Proceedings of the 2007 IEEE/RSJ International
Conference on Intelligent Robots and Systems
San Diego, CA, USA, Oct 29 - Nov 2, 2007

CASTRO: Robust Nonlinear Trajectory Optimization Using Multiple Models

Matthew McNaughton

Abstract— **In this paper we present CASTRO, a new approach for achieving robust planned trajectories for nonlinear systems in the presence of modelling uncertainty. With CASTRO, we simultaneously optimize trajectories for multiple copies of the same system model, each using different estimates for the system parameters. The systems are constrained to use the same policy. With an appropriate sampling of system parameters in the optimization problem, the trajectory will be robust when run on the real system, compared to a trajectory optimized with just one model. We present results on a simulated double-link pendulum swing-up problem.**

I. INTRODUCTION

Optimization is a powerful tool for finding trajectories that maximize a particular performance criteria, but it is prone to creating plans that are narrowly tuned to model parameter values. We are interested in computing optimal trajectories for nonlinear systems, with a particular focus on robot walking. The chosen trajectory should be robust to modelling errors in the system dynamics and to disturbances from the environment. We propose a planning formulation called CASTRO (Control-Augmented System for Trajectories, Robust and Optimal) to increase robustness in trajectory optimization. Major sources of uncertainty that affect planning and control in general include

- Parametric uncertainty. For example, an uncertainty in system mass results in correlated errors.
- Unmodelled dynamics. For example, unmodelled vibrational modes. These can be modelled as random noise, or by augmenting the model structure.
- Random disturbances, which can be modelled as random noise.

Random, uncorrelated (a.k.a. white) process and sensor noise can be handled with state estimation techniques such as Kalman filters. Parametric error, however, is more difficult for robot controllers to handle, since errors are correlated with previous states and actions. In the present work, we are interested in planning with representations of parametric uncertainty.

We would like to overcome several sources of modelling error with our Sarcos Primus System, a hydraulic humanoid. We have unknown hydraulic actuator dynamics which can result in errors in commanded joint torques. There are unknown, and difficult to estimate inertial parameters due to hoses and other flexible elements on the robot.

With CASTRO, several realizations of the system model are constructed using sampled values from the distribution of the unknown system parameters, and composed into a single large dynamic system. All of the component models are constrained to use the same policy, and a trajectory optimizer is used to find a trajectory that minimizes the expected cost for all of the component models to follow the trajectory. CASTRO does not require that modelling errors be treated as white noise. We validate CASTRO using a simulated double-link pendulum swing-up problem and compare results to a problem formulation that treats modelling error as white noise in the joint actuation.

The remainder of this paper is organized as follows. In Section II we discuss related work in planning and control under uncertainty. In Section III we describe CASTRO in more detail. In Section IV we present experiments with CASTRO. Sections V and VI finish with discussion and conclusions.

II. RELATED WORK

The literature on biped gait planning and robust planning and control is extensive. We provide a limited number of references here.

Robustness can be approached as a minimax problem, where one attempts to find a trajectory that minimizes the maximum cost for any model to follow the trajectory. For robust minimax optimization of linear systems, Boltyanski and Poznyak [2] prove conditions under which a control trajectory for a family of models can be found that minimizes the maximum time taken by any single model to reach a goal. Unfortunately, the results only apply to linear systems that are guaranteed to eventually reach the goal when the control input is equal to zero. Further, for many applications a minimum time trajectory is not what is desired.

Uncertainty in the model can be approximated by uncertainty in the state or its derivatives. Morimoto and Atkeson [9] [10] employed Differential Dynamic Programming (DDP), a trajectory optimization technique [7], using a minimax criterion to generate robust walking gaits for a five-link planar biped. In their approach, a single robot model is used, and an adversary is assumed to be applying unknown friction at the joints and an unknown disturbance torque at the ankle.

Thrun [19] provides an overview of particle filters, a popular and effective method for representing uncertainty in belief about a system's state. Typical applications use particle

This material is based upon work supported in part by the DARPA Learning Locomotion Program and the National Science Foundation under NSF Grants CNS-0224419, ECS-0325383, and EEC-0540865. M. McNaughton is supported by the Natural Sciences and Engineering Research Council of Canada PGS-D3 award. M. McNaughton is with the Robotics Institute, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, Pennsylvania, 15213 USA mmcnaugh@ri.cmu.edu

filters to construct a belief state about the system based on previous observations and actions but do not use them explicitly in the planning process. Thrun [18] uses particle filters in a Monte Carlo method to plan for POMDPs with real-valued state and action spaces. The result is Q-learning over belief states represented as particles. Copies of a system can be represented and propagated through time under the action of a single policy, and the robustness of the policy can be evaluated by the number of copies that reach the goal. Fu et al. [4] demonstrated a particle filter method that can select actions that reduce uncertainty in the system state.

Ng and Jordan [12] describe PEGASUS, which uses a gradient descent search in the parameter space of a parametric policy to find a policy that scores well on many randomized trials of a problem. The gradient of the collective score of the randomized trials is made continuous by providing the same sequence of random numbers to the system simulator for each set of trials. The present work is related, but while PEGASUS models uncertainty as uncorrelated noise, CASTRO uses parametric uncertainty. The many randomized trials of PEGASUS are replaced by the smaller number of models used in the trajectory optimization of CASTRO, and CASTRO policies are formulated as a trajectory with a linear controller in the neighbourhood of the trajectory.

Optimization based on maximizing a value function can tend to push the plan towards states with greater uncertainty. Bagnell et al. [1] describe Policy Search by Dynamic Programming (PSDP), which avoids this problem by backing up policies rather than values in the recursion.

There is work in the controls literature on the use of multiple model techniques to tackle complex control problems. The main thrust of these techniques is to compose simple controllers applicable in limited operating ranges of a plant into a single global controller usable over the whole operating range of the plant. An overview of the field can be found in [11].

We are using the trajectory optimizer DIRCOL [16] to create robot trajectories. There is previous work in using DIRCOL to create biped gait trajectories. Hardt [6] derived a cyclic gait for a five-link planar biped. Denk and Schmidt [3] precomputed a library of stepping trajectories for the robot Johnnie [8] and reported initial results on the robot.

III. Robust Trajectory Optimization using a Controller

A. Two-link pendulum swing-up trajectories

We explore CASTRO using the two-link pendulum model (Fig. 1). The dynamics are

$$\tau = M(q,\dot{q})\ddot{q} + C(q,\dot{q})\dot{q} + G(q),$$

where

$$q = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix},$$

and M, C, G are state-dependent matrices. Links are modelled as thin rods of uniform density. We used the Lagrange method to derive the equations of motion. We assume a massless torque source positioned at each joint.

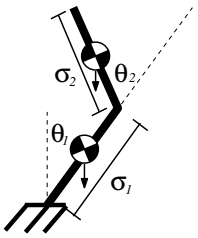

Fig. 1. The two-link pendulum.

TABLE I
Variables in the swing-up problem formulation

Symbol	Role
$t_0 = 0$	The initial time of the trajectory.
$t_f \leq 6s$	The time by which the swing-up trajectory must be complete.
$\theta_1(t)$, $\theta_2(t)$	Angular coordinates of the pendulum links.
$\dot{\theta}_1(t)$, $\dot{\theta}_2(t)$	Angular velocities of links
$\ddot{\theta}_1(t)$, $\ddot{\theta}_2(t)$	Angular accelerations of links.
τ_1, τ_2	The motor torques applied to the first and second links.
v	Viscous friction applied at each joint. $v = 0.1\dot{\theta}$ unless specified otherwise.
g	Gravitational acceleration. $g = 10$ in all experiments.

We use a version of the well known swing-up problem. Table I describes the system variables and Table II describes the boundary conditions on the variables at the start and end of the trajectory.

We define an optimal trajectory as one minimizing

$$\int_0^{t_f} \tau_1(t)^2 + \tau_2(t)^2 + 0.01(\theta_1(t)^2 + \theta_2(t)^2)dt. \quad (1)$$

We denote the state of the system by x, the control inputs (commanded motor torques) by u, the time by t, and the model parameters by p. They are defined as

$$x = \begin{bmatrix} q \\ \dot{q} \end{bmatrix} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix},$$

$$u = \begin{bmatrix} \tau \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix},$$

and

$$\dot{x} = \begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix} = \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix} = f(x,u,p,t).$$

TABLE II
Boundary conditions in the swing-up problem formulation

$\theta_1(0) = \pi$
$\theta_2(0) = \dot{\theta}_1(0) = \dot{\theta}_2(0) = 0$
$\theta_1(t_f) = \theta_2(t_f) = \dot{\theta}_1(t_f) = \dot{\theta}_2(t_f) = 0$

The cost function is written

$$\min \int x^T Q x + u^T R u,$$

where Q and R are symmetric positive definite, in this case $Q = 0.01I$ and $R = I$. I is the identity matrix.

A plan returned by a trajectory optimizer is a set of state and control histories x_d, u_d that satisfy the system dynamics $\dot{x}_d(t) = f(x_d, u_d, p, t)$ and any other constraints.

For our exploration, we are interested in plans where the link parameters are uncertain. Our links are thin rods with a single parameter σ describing the shape as

$$\begin{aligned} \text{mass}(m) &= \sigma, \\ \text{length}(l) &= \sigma, \\ \text{and} \quad \text{inertia} &= \tfrac{1}{12} m l^2. \end{aligned}$$

For the two-link pendulum, the links are described by shape parameters σ_1, σ_2, and we constrain

$$1 = \sigma_1 + \sigma_2$$

so that we have a single shape parameter describing the two-link pendulum

$$s = \sigma_1 - \sigma_2, -1 < s < 1.$$

B. Method

A reasonable planning method for robot motions when system parameters are uncertain is to choose nominal values for the unknown parameters, create a plan based on those values, and then apply feedback control when running the plan on the robot. That is, the control law applied to the robot is

$$u(t) = u_d(t) + K(x_d(t) - x(t)),$$

where K is a matrix of gains. The question that arises is whether the plan is reasonable when the parameters of the actual robot differ from the model used in planning. With CASTRO, the system state vector is augmented with additional copies of the model state, using alternate approximations of the system parameters. For the case of two models, the overall state becomes

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix},$$

where

$$\begin{aligned} x_1 &= \begin{bmatrix} q_1 \\ \dot{q}_1 \end{bmatrix} = \begin{bmatrix} \theta_{11} & \theta_{21} & \dot{\theta}_{11} & \dot{\theta}_{21} \end{bmatrix}^T, \\ \text{and} \quad x_2 &= \begin{bmatrix} q_2 \\ \dot{q}_2 \end{bmatrix} = \begin{bmatrix} \theta_{12} & \theta_{22} & \dot{\theta}_{12} & \dot{\theta}_{22} \end{bmatrix}^T, \end{aligned}$$

and the system dynamics are

$$f(x, u_1, p, t) = \begin{bmatrix} f_1(x_1, u_1, s_1, t) \\ f_1(x_2, u_1 + K(x_1 - x_2), s_2, t) \end{bmatrix},$$

with f_1 equal to the dynamics of the double-link pendulum, s_1 and s_2 the shape parameters of the first and second pendulum models,

$$u_1 = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix},$$

TABLE III
LIST OF EXPERIMENTS

Name	Description
S0	Single-model plan with shape $s = 0$.
Sp2	Single-model plan with shape $s = 0.2$.
Sm2	Single-model plan with shape $s = -0.2$.
C22	CASTRO plan using two models with shapes $s = -0.2$ and $s = 0.2$.
SCAL5	CASTRO scalability test with five models.
SCAL9	CASTRO scalability test with nine models.
SCAL20	CASTRO scalability test with twenty models, all $s = 0$
NOISE5	Uncorrelated noise test with five models.

and K is as described below. Note that the control input vector has not been augmented – only $u_1(t)$, the feedforward torque vector for the first model is provided. The cost function to be minimized is

$$\min \int_0^{t_f} x^T Q x + U(x, u)^T R U(x, u)$$

with $Q = 0.01I$, $R = I$, and

$$U(x, u) = \begin{bmatrix} u_1 \\ u_1 + K(x_1 - x_2) \end{bmatrix}.$$

We use the LQR method [15] to design a controller K. A and B of the LQR design process are formed by linearizing the dynamics of a pendulum with $s = 0$ about the goal(upright) position. We select Q and R matrices to produce gains stiff enough to regulate pendulums with any shape value in the range $-0.6 \leq s \leq 0.6$ from the starting state (hanging straight down) to the goal state. For Q and R we used diagonal matrices with $\text{diag}(Q) = \begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix}^T$ and $\text{diag}(R) = \begin{bmatrix} 0.1 & 0.1 & 0 & 0 \end{bmatrix}^T$ resulting in a 2×4 matrix of gains,

$$K = \begin{bmatrix} 11.0 & 2.2 & 2.6 & 0.8 \\ 1.9 & 4.8 & 0.7 & 0.4 \end{bmatrix}.$$

IV. EXPERIMENTS

All experiments were run on a quad-processor 3GHz Intel Xeon with 2GB of system RAM. We used DIRCOL[16] with SNOPT[5] as a back-end nonlinear minimization solver using the sparse Jacobian.

A. Single-Model Plans

As a baseline, we derived an optimal trajectory for a two-link pendulum with shape parameter $s = 0$ and the cost function and conditions specified in the previous section (Experiment S0, see Table III). Fig. 2 shows an animation illustrating the trajectory of the pendulum. It makes a small pump bringing the elbow left, then pumps hard to lift the proximal link above the shoulder, finally swinging the distal link straight up.

The trajectory was applied in simulation using this controller to fifteen pendulums with shape parameters in the range $-0.6 \leq s \leq 0.6$. Integration along the trajectory was done with the `ode15s` function of MATLAB[17]. We measured the cost for each pendulum to follow the trajectory from Equation 1. The costs are shown in Fig. 3.

Fig. 2. Single-model plan for a two-link pendulum with shape parameter 0 (Experiment S0).

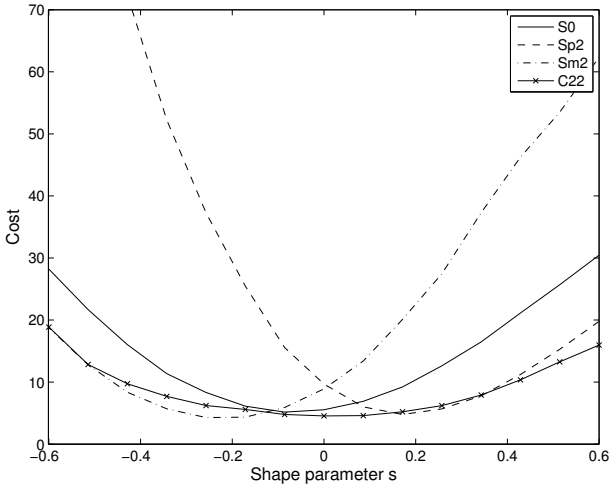

Fig. 3. Costs of trajectories S0, Sp2, Sm2, and the CASTRO trajectory C22

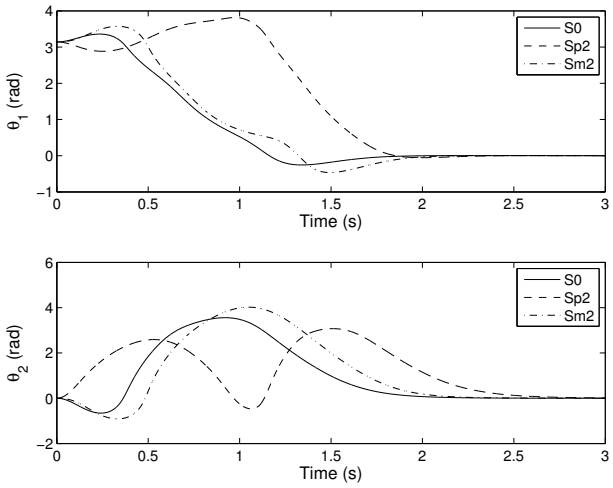

Fig. 4. Kinematics of trajectories planned in Experiments S0, Sp2, and Sm2.

We also derived trajectories for $s = 0.2$ (Experiment Sp2) and $s = -0.2$ (Experiment Sm2). They had a similar character to S0. Traces of the θ_1 and θ_2 are compared in Fig. 4. Trajectories S0 and Sm2 are very similar in overall shape, while Sp2 has an extra pump.

These trajectories were applied using the same controller to the same range of pendulum shapes as above. The costs are compared in Fig. 3. Notice that each of the trajectories, while of comparable cost to each other at their respective nominal s-values, are not low even over the full range of $-0.2 \leq s \leq 0.2$, and can increase quite dramatically as the shape parameter goes outside that range. We were surprised to see that the minima of the cost curves are not at the shape values used to create the plans. The time required to compute each trajectory was a few minutes.

TABLE IV

PARAMETERS OF NINE MODELS IN SCALABILITY TEST

s (shape)	v (friction)
0	0.1
0.1	0.1
0.2	0.1
0.2	0.2
0.3	0.1
-0.3	0.1
-0.2	0.2
-0.1	0.1

B. Experiments with CASTRO

We put the two-link pendulum problem into a CASTRO formulation using a total of two models. The first (reference) model with $s = -0.2$, and the second, controlled model with $s = 0.2$. We used the same matrix K of gains as with the experiments of the previous section. An animation of the trajectory found by the planner (Experiment C22) can be seen in Fig. 5. The pendulum first pumps the elbow left, then right. It lifts the distal link, pumps once with the distal link lifted up and the proximal link still hanging down, then swings the proximal link up, followed by the distal link.

It is interesting to compare the cost of the CASTRO trajectory C22 to the costs of the trajectories S0, Sp2, and Sm2 that were planned using a single pendulum model. Fig. 3 shows that C22 has a cost comparable to the minima of all three of those trajectories. Note that the CASTRO trajectory is cheaper for $s = 0$ than was S0. This indicates that the planner was stuck in a local rather than global minimum, a common problem. C22 required a few minutes of CPU time to compute.

C. Scalability

DIRCOL is able to solve problems of high dimensionality. We have successfully run a CASTRO formulation of the two-link pendulum swing-up problem using nine component models (Experiment SCAL9), requiring a total of 37 state variables. There are four state variables required for each of the nine copies, plus one for cost. The model parameters chosen for each model are shown in Table IV. The values were not chosen to reflect any particular underlying distribution. A good initial guess was taken from trajectory C22. The maximum memory usage was approximately 100 megabytes and the time required was 1.5 hours. In practice, it is likely not possible for the planner to converge on a solution for a large CASTRO system such as this without the use of a good

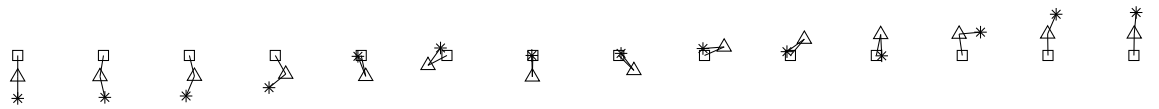

Fig. 5. Plan for CASTRO system with $s = 0.2, -0.2$ (Experiment C22).

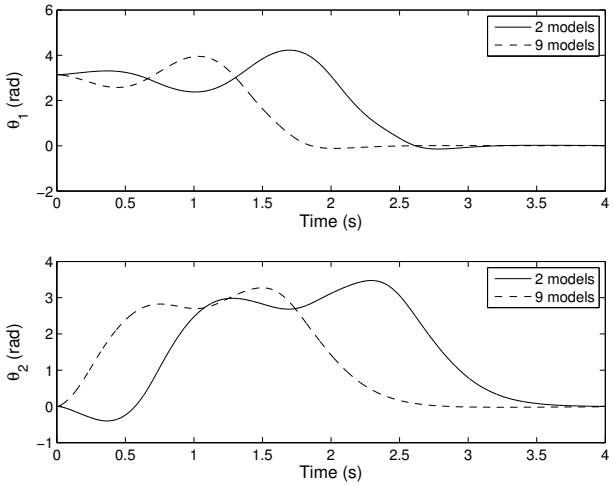

Fig. 6. Kinematics of trajectories C22 vs. SCAL9

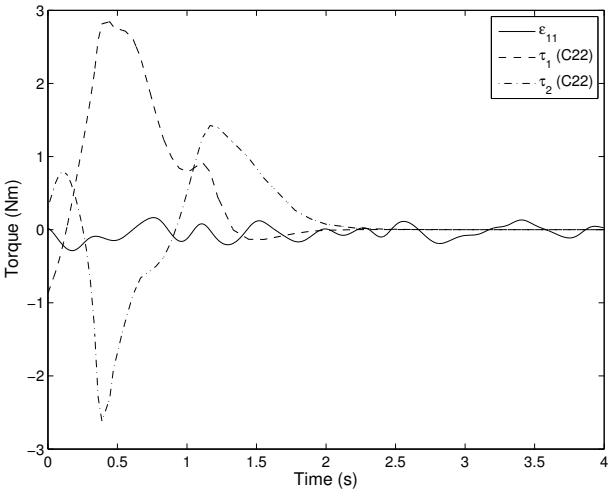

Fig. 7. Shape and magnitude of noise signal ϵ compared to the noise-free feedforward torque from trajectory C22.

initial guess obtained from a single-model system. Fig. 6 shows the resulting trajectory as similar in overall form to C22, with three pumps rather than four in θ_1, and one fewer pump in θ_2. Fig. 9 shows the cost of this trajectory when run on pendulums of various shapes, compared to those of Fig. 3.

D. Uncorrelated noise

CASTRO approximates the effects of correlated noise caused by uncertainty in system parameters by creating representative models. One can also use CASTRO to approximate uncertainty using uncorrelated (white) noise. PEGASUS is an example of this approach. To compare the effects of modelling uncorrelated versus correlated noise on the plan, we formulated a CASTRO problem using five identical models, all with $s = 0$ (Experiment NOISE5). The motor torque applied to each joint of each model was corrupted with a unique sequence of random noise. The whole system dynamics are

$$f(x, u_1, p, t) = \begin{bmatrix} f_1(x_1, u_1, p, t) \\ f_1(x_2, \epsilon_2(t) + u_1 + K(x_1 - x_2), p, t) \\ f_1(x_3, \epsilon_3(t) + u_1 + K(x_1 - x_3), p, t) \\ f_1(x_4, \epsilon_4(t) + u_1 + K(x_1 - x_4), p, t) \\ f_1(x_5, \epsilon_5(t) + u_1 + K(x_1 - x_5), p, t) \end{bmatrix},$$

where each $\epsilon_i(t)$ is a function composed of Gaussian kernel functions smoothing over N random numbers sampled at evenly-spaced times

$$\epsilon_i(t) = \begin{bmatrix} \eta \sum_{j=0}^{N-1} d_{ij1} \exp(\frac{-(j\sigma - t)^2}{\sigma^2}) \\ \eta \sum_{j=0}^{N-1} d_{ij2} \exp(\frac{-(j\sigma - t)^2}{\sigma^2}) \end{bmatrix},$$

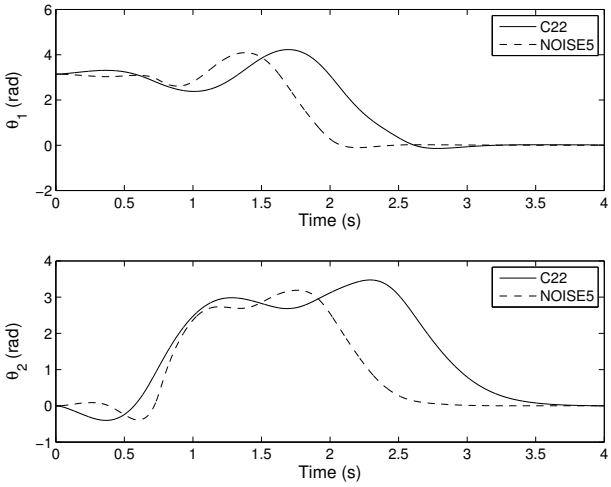

Fig. 8. Comparison of trajectory NOISE5 with C22

where η is an overall scaling factor, $d_{ijk} \sim \mathcal{N}(0, 1)$ are precomputed random numbers drawn from a Gaussian distribution with mean 0 and variance 1, and σ is the spacing in time between the noise samples d_{ijk}. For this experiment, we chose $\eta = 0.1$, $N = 60$, and $\sigma = 0.1$. Fig. 7 shows as an example the noise sequence for θ_2 of $\epsilon_2(t)$ compared to the feedforward torque signals in the trajectory of C22.

The run took four hours and converged to a locally optimal result. Fig. 8 shows the trajectory of Experiment NOISE5. Compared to trajectory C22, it has additional pumps at the beginning and completes in a shorter time. Fig. 9 compares the costs of this plan when run on pendulums with a range

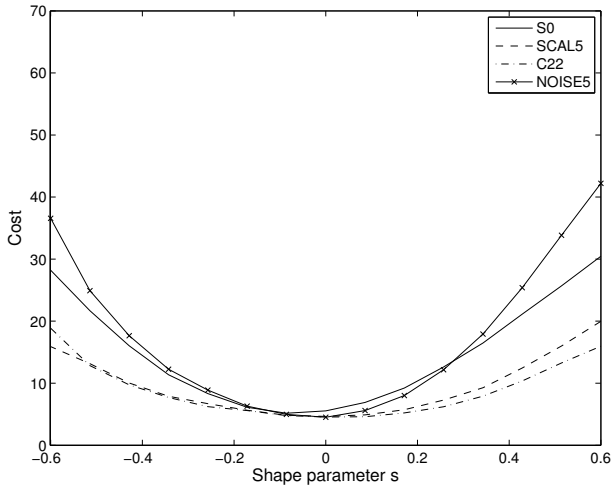

Fig. 9. Costs of SCAL9 and NOISE5 run on various pendulums compared to S0 and C22

of shapes to the costs of a plan made using a single model. The NOISE5 trajectory is the same cost as SCAL9 when $s = 0$, but the cost rises rapidly at values of s increasingly far from 0.

V. Discussion

Our experiments in the scalability of CASTRO showed that it continues to be effective up through nine component models. The planner was unable to converge to a solution using twenty component models. However, it is not certain that this shows a hard limit on the problem size addressable by CASTRO. Further tuning of the optimization parameters or the initial guess could result in success. CASTRO's reach will also increase with progress in the underlying constrained nonlinear minimization. CASTRO shows the benefits of using correlated noise, that is, planning with explicit representations of parametric uncertainty, compared to the alternative of approximating model uncertainty as uncorrelated, or "white" noise. We have also seen that we can benefit from integrating the controller directly into the planner. We can effectively plan for uncertainty in system model parameters using multiple representative copies of the model with different estimates of the parameter values.

VI. Conclusions

We have shown CASTRO to be an effective method for planning under uncertainty in model parameters with a fully actuated continuous system of modest dimensionality. Future work will include extending CASTRO to problems with discontinuities in system dynamics, walking in particular. We will experiment with underactuated systems such as the Acrobot[14] and Pendubot[13]. We will compare the effects of using alternate cost function structures, such as minimax cost functions. We are also interested in exploring the reach of CASTRO to a broad range of problems in optimal steering, such as UAVs and autonomous rovers. Future work will also include experiments with other methods of optimization, such as genetic algorithms.

VII. Acknowledgments

The author would like to thank his advisor Dr. Christopher G. Atkeson for guidance in this work and for reviewing drafts of the paper. The author also thanks Benjamin Stephens, Lillian Chang, and Vinithra Varadharajan for their comments on drafts of this paper.

References

[1] James Bagnell, Sham Kakade, Andrew Ng, and Jeff Schneider. Policy search by dynamic programming. In *Neural Information Processing Systems*, volume 16. MIT Press, December 2003.

[2] V. Boltyanski and A. Poznyak. Linear multi-model time-optimization. *Optimal Control Applications and Methods*, 23(3):141–161, 2002.

[3] J. Denk and G. Schmidt. Synthesis of walking primitive databases for biped robots in 3D-environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA2003)*, Taipei, Taiwan, 2003.

[4] Jiaxin Fu, Siddhartha Srinivasa, Nancy Pollard, and Bart Nabbe. Planar batting under shape, pose, and impact uncertainty. In *IEEE International Conference on Robotics and Automation (ICRA)*, April 2007.

[5] Philip E. Gill, Walter Murray, and Michael A. Saunders. Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM Journal on Optimization*, 12(4):979–1006, 2002.

[6] Michael Hardt. *Multibody Dynamical Algorithms, Numerical Optimal Control, with Detailed Studies in the Control of Jet Engine Compressors and Biped Walking*. PhD thesis, University of California San Diego, June 1999.

[7] David H. Jacobson and David Q. Mayne. *Differential Dynamic Programming*. American Elsevier Publishing Company, Inc., New York, NY, 1970.

[8] K. Löffler, M. Gienger, and F. Pfeiffer. Sensors and control concept of walking "johnnie". *The International Journal of Robotics Research*, 22(3-4):229–239, March–April 2003.

[9] Jun Morimoto and Christopher Atkeson. Minimax differential dynamic programming: An application to robust biped walking. In *Neural Information Processing Systems*, 2002. Vancouver, Canada.

[10] Jun Morimoto and Christopher Atkeson. Robust low torque biped walking using differential dynamic programming with a minimax criterion. In *CLAWAR*, September 2002.

[11] R. Murray-Smith and T. Johansen, editors. *Multiple Model Approaches to Modelling and Control*. CRC, March 19 1997.

[12] Andrew Y. Ng and Michael Jordan. PEGASUS: A policy search method for large MDPs and POMDPs. In *Uncertainty in Artificial Intelligence, Proceedings of the Sixteenth Conference*, 2000.

[13] M. W. Spong and D. J. Block. The pendubot: a mechatronic system for control research and education. In *Proceedings of the 34th IEEE Conference on Decision and Control*, pages 555–556, December 1995.

[14] M.W. Spong. The swing up control problem for the acrobot. *Control Systems Magazine*, 15(1):49–55, Feb 1995.

[15] Robert F. Stengel. *Stochastic Optimal Control: Theory and Application*. John Wiley & Sons, Inc., 1986.

[16] O. Stryk. Numerical solution of optimal control problems by direct collocation. In R. Bulirsch, A. Miele, J. Stoer, and K.-H. Well, editors, *Optimal Control, Calculus of Variations, Optimal Control Theory and Numerical Methods*, International Series of Numerical Mathematics 111. Birkhauser.

[17] The MathWorks, Inc., 3 Apple Hill Drive, Natick, MA, USA. MATLAB. http://www.mathworks.com.

[18] S. Thrun. Monte carlo POMDPs. In S.A. Solla, T.K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 1064–1070. MIT Press, 2000.

[19] S. Thrun. Particle filters in robotics. In *Proceedings of the 17th Annual Conference on Uncertainty in AI (UAI)*, 2002.