Dynamic Task Execution using Active Parameter Identification with the Baxter Research Robot

Andrew D. Wilson, *Student Member, IEEE,* Jarvis A. Schultz, *Member, IEEE,*
Alex R. Ansari, *Student Member, IEEE,* and Todd D. Murphey, *Member, IEEE*

***Abstract*—This paper presents experimental results from real-time parameter estimation of a system model and subsequent trajectory optimization for a dynamic task using the Baxter Research Robot from Rethink Robotics. An active estimator maximizing Fisher information is used in real-time with a closed-loop, non-linear control technique known as Sequential Action Control. Baxter is tasked with estimating the length of a string connected to a load suspended from the gripper with a load cell providing the single source of feedback to the estimator. Following the active estimation, a trajectory is generated using the `trep` software package that controls Baxter to dynamically swing a suspended load into a box. Several trials are presented with varying initial estimates showing that estimation is required to obtain adequate open-loop trajectories to complete the prescribed task. The result of one trial with and without the active estimation is also shown in the accompanying video.**

***Note to Practitioners*—This paper experimentally demonstrates the capability of an on-line parameter learning algorithm on the Baxter Research Robot to improve task performance. This type of algorithm could enable automated systems to actively inspect multi-body parts for parametric information including estimation of the robot's own inertias. The method requires known equations of motion for any nonlinear system with uncertain, constant parameters. We show using a series of 18 experimental trials that using the estimation method results in improved task performance for automated dynamical motions given uncertain parameters.**

***Index Terms*—parameter estimation, optimal control, maximum likelihood estimation**

I. Introduction

ONE fundamental goal of artificial learning for automation and production is providing the capability for a robot to automatically synthesize actions that improve estimates of the robot's internal dynamics and dynamic models of real-world objects. Human workers on production lines constantly use dynamic interactions with objects to improve their quality and speed in a manufacturing environment. We aim to provide this form of learning on robots using real-time processing of feedback from active exploration of the environment. Since the general problem of model synthesis and learning remains a formidable one, we restrict ourselves in this paper to creating a method for real-time active synthesis of dynamic trajectories to estimate a single model parameter in a known dynamical model.

A. Wilson, J. Schultz, A. Ansari, and T. Murphey are with the Department of Mechanical Engineering, Northwestern University, 2145 Sheridan Road, Evanston, IL 60208, USA. awilson@u.northwestern.edu, jschultz@northwestern.edu, alexanderansari2011@u.northwestern.edu, and t-murphey@northwestern.edu

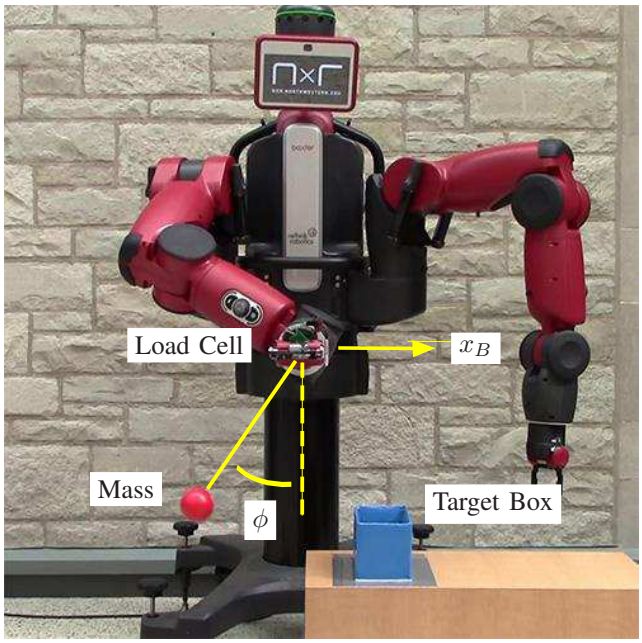


Fig. 1. Baxter performing real-time active parameter estimation.

Active estimation of parameters within dynamical systems, also referred to as *optimal experimental design*, commonly uses Fisher information as the primary metric [1]. Fisher information provides a best-case estimate of the estimator's performance given a set of measurements from a robot through the Cramer-Rao bound [2], [3]. A number of works on "exciting" trajectories by Armstrong and others [4]–[6] provide the theoretical basis for information-based estimation. In work by Emery [1], least-squares and maximum-likelihood estimation techniques are combined with Fisher information to optimize the experimental trajectories. In this case and several others, dynamics are solved as a discretized, constrained optimization problem [7]–[9]. One downside is that this time discretization can lead to high dimensional optimization problems (dimensions of 10^7 to 10^{12} are common in practice).

One question that may be raised by the reader is why a trajectory optimization algorithm is necessary for excitation. Non-algorithmic approaches such as frequency sweeps on the control inputs can be performed which will likely provide some level of Fisher information; however, the real cost to a sweep approach is a large use of energy in the control inputs. Using an optimization algorithm generates trajectories that provide an appropriate level of information with far

less control energy than a non-optimal excitation. Especially for under-actuated systems, exciting certain harmonics of the system and exploiting the free dynamics is critical to minimizing the control energy which is achieved through trajectory optimization.

This paper expands on preliminary results by the authors using Sequential Action Control for Fisher information maximization and parameter estimation. The preliminary results, presented at the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems [10] include the parameter estimation algorithm while this paper provides results from a practical task involving a dynamic system. Detailed derivations of the underlying control principles and information theory can be found in [11]–[14]. The Baxter Research Robot has been used as a practical platform for a number of studies [15]–[17] while also presenting a number of challenges including high compliance and actuator saturation. Despite these potential sources of unmodeled dynamic effects, results show that the estimator successfully converges to the actual parameter value across several trials and completes the dynamic task with the correct parameter estimates.

This paper is organized as follows: Section II provides an overview of the algorithmic foundation for the experiment. The specific implementation details for Baxter and dynamic task are provided in Section III. Section IV provides results from several trials of the real-time estimation task, and Section V concludes the paper with notes on future work. The estimation and dynamical task execution presented in this paper are also shown in the accompanying video.

II. Algorithmic Overview

This section presents an overview of the optimal control algorithms implemented in Section III. There are two stages to the control problem: First, an unknown parameter must be estimated in the system. For the example presented in this paper, the string length of the suspended mass is uncertain. Acquiring a better estimate will allow the optimized task trajectory to complete successfully; however, estimation of the parameter requires active exploration by the robot. The active estimation algorithm involves an extension of the Sequential Action Control (SAC) algorithm [12], [13].

After estimation, the second stage is the synthesis of the task trajectory. The optimization of the task trajectory is performed using a projection-based nonlinear trajectory optimization routine provided by `trep`, a simulation and optimal control package available at `http://nxr.northwestern.edu/trep`. Since there is no state feedback in this example, the task trajectory is run open-loop which requires accurate model parameters obtained from the first stage. The following sections provide detail on the two control stages.

A. Active Parameter Estimation

As shown in Fig. 2, there are two primary modules interacting with the robot hardware: the SAC controller for trajectory synthesis and the nonlinear least-squares estimator. At the highest level, the least-squares estimator requires the control inputs provided by the SAC controller to compute a predicted output which is compared to the actual measurements provided by the robot. The SAC controller is updated with new parameter estimates and state estimates by the least-squares estimator and optionally can receive state feedback directly from the robot hardware if available. These modules can be run asynchronously and at different rates with the use of a nonlinear state observer model.

In this paper, we assume that one parameter is uncertain with additive noise on observer measurements but negligible process noise. The same cost function with several unknown parameters is presented in [11]. For SAC, the state is usually derived assuming control-affine dynamics [12]. Thus, the model of the system is defined as

$$\begin{aligned} \dot{x} =& f(x,u,\theta) = g(x,\theta) + h(x,\theta)u \\ \tilde{y} =& y(x,u,\theta) + \mathrm{w}_y \end{aligned} \tag{1}$$

where $x \in \mathbb{R}^n$ defines the system states, $\tilde{y} \in \mathbb{R}^h$ defines the measured outputs, $u \in \mathbb{R}^m$ defines the inputs to the system, $\theta \in \mathbb{R}$ defines the parameter to be estimated, and w_y is additive output noise where $p(\mathrm{w}_y) = N(0,\Sigma)$.

In order to maximize information, the SAC cost function is modified to include a cost on the Fisher information of the uncertain parameter. For this implementation, we assume that the measurement noise of the system is normally distributed with zero process noise. Therefore the Fisher information is given by

$$I(\theta) = \sum_{k=k_0}^{k_f} \Gamma_\theta(t_k)^T \cdot \Sigma^{-1} \cdot \Gamma_\theta(t_k) \tag{2}$$

where Γ_θ is the derivative of the output y w.r.t. the parameter θ given by

$$\begin{aligned} \Gamma_\theta(t_i) =& D_x y(x(t_i),u(t_i),\theta) \cdot D_\theta x(x(t_i),u(t_i),\theta) \\ &+ D_\theta y(x(t_i),u(t_i),\theta). \end{aligned}$$

As detailed in [11], a cost function on Fisher information from (2) requires the simulation of the gradient of x w.r.t. θ, i.e. $\psi(t) = D_\theta x$. These additional states are referred to in the paper as extended state dynamics and notated as $\bar{x} = (x,\psi)$. For this implementation, we use only a running cost for the Sequential Action Controller which is written as

$$J_\tau = \int_{t_0}^{t_f} l(\bar{x}(t))\, dt \tag{3}$$

where

$$l(\bar{x}(t)) = \left[\Gamma_\theta(t)^T \cdot \Sigma^{-1} \cdot \Gamma_\theta(t)\right]^{-1} + x(t)^T \cdot Q_\tau \cdot x(t)$$

i.e., the minimization of the inverse of the information and an optional trajectory tracking cost to bias the system toward a particular part of the state space.

The SAC control synthesis process follows a receding-horizon style format to sequence together separately short optimal control *actions* into a piecewise continuous constrained feedback response to state, similar to a nonlinear model predictive controller but with a single control action [18], [19]. In SAC, *actions* are defined by a pair composed of a control

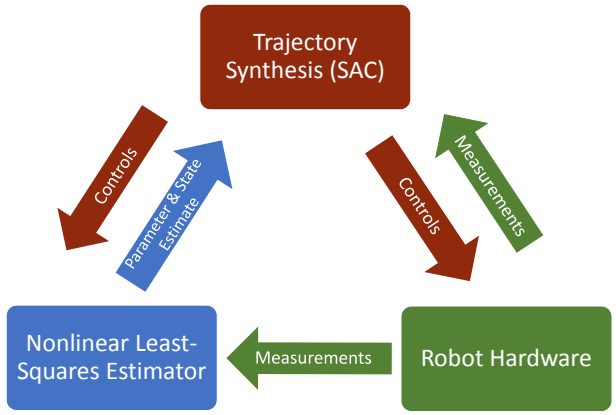


Fig. 2. Overview of the SAC active estimator real-time control structure.

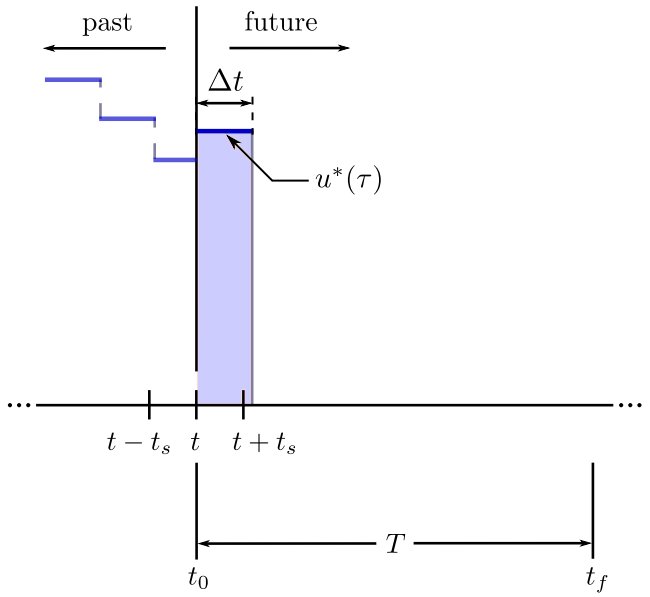


Fig. 3. SAC actions for a 1-D control are sequenced in receding-horizon fashion.

vector value and its associated (typ. short) application duration. The blue shaded region in Fig. 3 shows a SAC action for a 1-D control where $u^*(\tau)$ is the control and Δt is the duration.

The SAC algorithm predicts system motion from current state feedback, $x(t_0) = x_0$. The process involves simulation of a state and adjoint system (x, ρ) for a fixed time horizon, T, until (receding) final time $t_f = T + t_0$. For the purposes of this paper, the nominal control value used for simulations (x, ρ) is $u = 0$ so that SAC computes optimal actions relative to the free (unforced) system motion.

The adjoint variable $\rho : \mathbb{R} \mapsto \mathbb{R}^{2n}$ provides information about the sensitivity of the cost function to the extended state, $\bar{x}$. The algorithm maps this sensitivity to a control sensitivity provided by an inner product between the adjoint and dynamics (1). The process of control synthesis uses this sensitivity, $\frac{dJ_\tau}{du_\tau}$, to search for least norm actions that optimize the expected change in cost (3). Thus SAC actions optimize the rate of trajectory improvement. These optimal actions depend directly on the adjoint, which is determined from open-loop simulation of the following equation,

$$\dot{\rho} = -D_{\bar{x}} l(\bar{x})^T - D_{\bar{x}} f(\bar{x}, u)^T \rho$$

with a terminal condition $\rho(t_f) = 0$. The adjoint equation is evaluated along the nominal system trajectory with $u = 0$. The ability to calculate an optimal action at the current time from a single adjoint differential equation enables the use of the algorithm for real-time computation. For a complete derivation of SAC control synthesis with examples see [12].

While the robot is executing a motion, a nonlinear least-squares estimator is used on-line to update the estimated value of the parameter as well as the robot state.

The least-squares estimator can be written as

$$\hat{\theta} = \arg\min_{\theta} \beta(\theta) \tag{4}$$

where

$$\beta(\theta) = \frac{1}{2} \sum_{i}^{h} (\tilde{y}(t_i) - y(t_i))^T \cdot \Sigma^{-1} \cdot (\tilde{y}(t_i) - y(t_i)). \tag{5}$$

$\tilde{y}(t_i)$ is the observed state at the i^{th} index of h measurements, $\Sigma \in \mathbb{R}^{h \times h}$ is the covariance matrix associated with the sensor measurement error, and $\hat{\theta}$ is the least-squares estimate of the parameter. The estimator recomputes a new estimate at a set frequency, incorporating any new measurements received since the last iteration.

Given this estimator, we will use gradient descent with a backtracking line-search to find optimal parameter values by minimizing the least-squares error in (5). Since the estimator requires a predicted output $y(t)$, which is based on current estimates of θ and $x(t)$, a nonlinear state observer is also required for systems without full-state feedback. While any numerical differential equation solver may be sufficient, for this implementation, we use the `trep` software package, which is also used for the task trajectory optimization detailed in the following section.

B. Task Trajectory Optimization

Once the uncertain parameter has been accurately identified using the SAC active estimator, the task trajectory must be synthesized. At the end of the estimation process, Baxter's end effector is returned to the zero position and the motion of the mass is allowed to naturally dampen while the task trajectory computation completes. This process can be seen in the accompanying video.

The selection of an appropriate optimal control method is dependent both on the computational requirements and level of dynamic motion in the task. Sequential Action Control was selected for the on-line estimator since real-time performance was desired. To achieve real-time performance, the SAC algorithm computes the current control over a long time horizon; however, future controls are not taken into account when computing an optimal action. For the estimator, this is acceptable since the model parameters are incorrect, and future controls are largely irrelevant relative to the measurement feedback. Once the parameters are known, a complete trajectory optimization method can then be used.

For this paper, optimization of the task trajectory is completed using the `trep` software package which implements a discrete mechanics version of projection-based nonlinear optimal control [20], [21]. This method provides a trajectory which is locally optimal w.r.t. the initial trajectory provided. For this example, a box is placed 0.45m to the left of the suspended mass. The dynamic task is to swing the mass over the box so that it lands inside the box and the initial trajectory is simply the zero trajectory.

The objective function for the task trajectory uses a terminal cost on the nominal states of the suspended mass and a running cost on the control given by

$$\begin{aligned} J_{\text{task}} =& (x(t_f) - x_d(t_f))^T \cdot P_\tau \cdot (x(t_f) - x_d(t_f)) \\ &+ \int_{t_0}^{t_f} u(t)^T \cdot R_\tau \cdot u(t)\, dt, \end{aligned} \tag{6}$$

where P_τ and R_τ are the state and control weights.

Each step of the projection-based optimization algorithm returns a new dynamically feasible trajectory with an improved cost. The software uses discrete-time algebraic Riccati equations resulting from a LQ problem formulation to produce a perturbation to the current trajectory which is then projected to a feasible curve. This process is repeated iteratively until the magnitude of the cost derivative is below a specified tolerance. For the results presented in this paper, the tolerance is set to 10^{-6}.

III. Baxter Implementation

This section describes both the problem formulation and software specific implementation of the control structure described in the previous section on the Baxter Research Robot created by Rethink Robotics [22]. Results from experimental trials are presented in Section IV.

A. Problem Description and Model Formulation

To test the learning capabilities on a practical robotic system, we created a dynamical task which involves swinging a suspended mass into a nearby box. In order to require the use of dynamics to solve the problem, we restrict Baxter's end effector in software to only move along the horizontal axis x_B; therefore, the mass must be swung to land inside the box shown in Fig. 1. However, we assume that the string length of the suspended mass is uncertain and must be estimated prior to optimizing the task trajectory.

As there is no feedback on the angle of the suspended mass, a load cell mounted at Baxter's end effector is used as the sole output to the estimator. This configuration necessitates active motion to estimate the string length as the Fisher information is zero when the robot is stationary. To simplify the control of Baxter, we chose to approximate the end effector as kinematic and control motion only in the Cartesian x-axis, x_B. The equations of motion for the active estimation stage are given by the following,

$$f_{\text{SAC}}(x, u, \theta) = \begin{bmatrix} \dot{x}_B \\ u \\ \dot{\phi} \\ \frac{u}{\ell}\cos\phi - \frac{g}{\ell}\sin\phi \end{bmatrix}$$

where u is the x-axis acceleration of the gripper, m is the known mass suspended from the robot, and ℓ is the length of the string, which will be estimated. Additionally, the equation for the force output F_s is

$$y(x, u, \theta) = F_s = mg\cos\phi - m\ell\dot{\phi}^2 - u\sin\phi.$$

It is assumed that the trajectories will maintain tension in the string; therefore, the distance between the robot and mass is fixed. Given this system model, the extended state $\bar{x} \in \mathbb{R}^8$.

After the first estimation stage, a slightly different choice of states is made for the task trajectory in the second control stage. Since a terminal cost is given as a function of the Cartesian position of the suspended mass in the vertical x-z plane, those will be the system states along with the position of Baxter's end effector x_B. A distance constraint is added in the `trep` software which defines a constant distance between the end effector and mass which represents the string length.

B. Experimental Implementation

As shown in Fig. 2, there are essentially four modules interacting with the robot hardware: the SAC trajectory synthesis module, a measurement module, the nonlinear least-squares estimator module, and the task trajectory module. In this section we describe how each of these modules is implemented for the Baxter experiments.

The backbone of communication between modules and the robot is provided by the Robot Operating System (ROS) [23]. ROS provides the ability to asynchronously run the control and estimation modules, implemented as ROS nodes, and Baxter natively uses ROS as the primary API for motion commands. Three primary nodes are employed: the SAC control node, a measurement receiver node, and the estimator node.

1) Baxter SAC Node: As discussed in the previous section, the end effector of the Baxter robot is approximated as a kinematic input to the dynamic suspended mass system. The dynamic model for the suspended mass system assumes only planar motion, and the kinematic input moves perpendicular to gravity in this plane. It follows that the SAC module produces target locations along a one-dimensional line that Baxter's end effector should follow. A joint velocity controller for Baxter's right arm is used to stabilize the right end effector to these target locations. To avoid issues with kinematic singularities in the manipulator while controlling to these target locations, this one-dimensional line is expanded to a candidate set of closely-spaced end effector targets in $SE(3)$. An off-line computation is done to solve the inverse kinematics problem for each of the targets in the set producing a set of target joint angles for each target. These joint angle targets are then stored to disk as a lookup table between target horizontal positions, like those produced by SAC, and target joint angles. During an experimental run, the desired joint position, velocity, and acceleration data is sent using a Joint Trajectory Publisher to Baxter's internal controller which runs a high frequency real-time loop to control each of the joints.

2) Baxter Measurement Receiver Node: The payload is attached to Baxter's end effector through a one-dimensional load cell. A microcontroller samples the load cell at 100 Hz

and transmits the measured forces via a serial link back to a client computer which is communicating with Baxter over an Ethernet connection. These force measurements represent the $\tilde{y}(t_i)$ in (5). The load cell has been calibrated prior to the experiment to convert the load cell output to a force (N). The resulting force is timestamped as it is received and published at 100 Hz for use by the estimator node.

3) Baxter Estimator Node: The estimator node subscribes to the actual end effector trajectory which is provided by the Baxter API. These measurements are used to generate the $y(t_i)$ terms in (5) through the use of a nonlinear state observer. As mentioned in Section II-A, the `trep` software package is used as the state observer. The trajectory is used as the input and `trep` provides the predicted state evolution of the suspended load. From these states, the predicted force, $y(t)$ can be calculated.

At a frequency of 2 Hz, the estimation module solves the optimization problem described by (4), updating the estimate of the string length. This frequency was chosen as a conservative rate at which the estimator has enough time to provide an update; however, the rate can be modified depending on the required computational time for the particular system. This parameter estimate and new state estimate are provided as a service to the SAC node which queries the service at the start of each computation. Updates to the string length estimate and expected state are reflected in this service call.

4) Baxter Task Node: The previous three nodes are all used for the estimator in the first control stage. Following the estimation, a task node is created that uses `trep` to compute an optimal task trajectory using the estimated value of the string length. `trep` uses a discrete timestep of 0.01s for the numerical simulation resulting in a control frequency of 100Hz for the task execution.

Position, velocities, and accelerations are sent open-loop to the Baxter robot using a Joint Trajectory Publisher in ROS. The same lookup table as in the SAC node is used to generate joint positions and a corresponding Jacobian table is used for velocities. Accelerations are provided using a finite difference of the joint velocities. A finite-time horizon of 5.0 s is used for the task trajectory. At the end of the trajectory, the end effector is commanded to a position over the box that allows the suspended load to fall directly into the box.

IV. Experimental Results

This section presents the results of the experimental trials of the algorithms using the Baxter robot. To compare the ultimate goal of task performance given uncertain estimates of the string length of the suspended mass, a total of 18 trials were run. The first set of 9 trials were run with varying initial length estimates, and these parameter values were directly used in the task trajectory optimization without the active estimation stage. The second set of 9 trials were run with the same distribution of initial estimates; however the first active estimation stage was run to attempt to provide a more accurate value of the string length for the task optimization.

TABLE I
Experimental Task Results

Initial Length Estimate (m):	Without SAC Estimation:	With SAC Estimation:
0.308	Fail	Success
0.328	Fail	Success
0.348	Fail	Success
0.368*	Success	Success
0.388	Success	Success
0.408	Fail	Success
0.428	Fail	Success
0.448	Fail	Success
0.468	Fail	Success

*actual string length

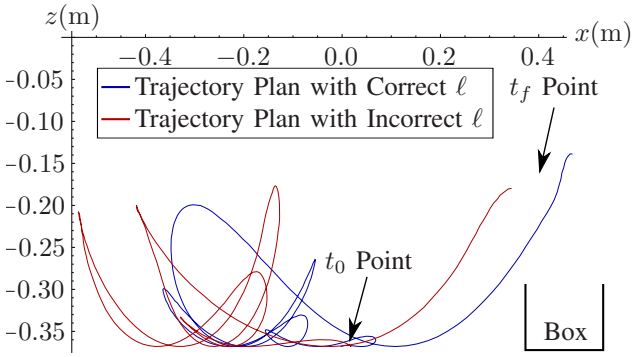


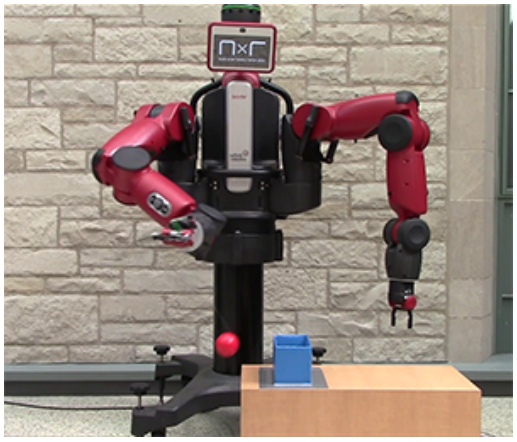
Fig. 4. Simulated trajectory plans for the suspended mass with the correct $\ell = 0.368$m and incorrect $\ell = 0.328$m.

A. Direct Task Optimization Results

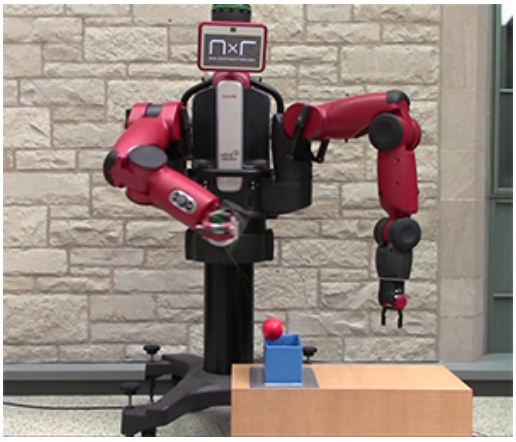
To evaluate the performance of the Baxter robot with an open-loop task and uncertain parametric information, the first 9 trials were run using the initial length estimates given in the first column of Table I. Using `trep`, a trajectory was synthesized using a parameter value from the table and the task was attempted. If the suspended mass landed inside the box at the end of the trial, the trial is considered a success. Any trial with the mass outside of the box at the end of the trial resulted in failure.

The results of these 9 trials are given in the center column of Table I. The task is completed with only two estimates of the length - the actual value of 0.368m and a slightly longer length of 0.388m. Figure 4 shows the difference between one path of the suspended mass simulated at correct value of 0.368m and the simulated path planned at an incorrect value of 0.328m. As shown in the figure, the incorrect parameter value results in motion which does not swing the mass far enough over to land above the box. This difference is also highlighted in frame captures from the experimental trials in Fig. 5. The experimental frame captures mirror the end of the simulated trajectories from Fig. 4 with the incorrectly planned trajectory failing to swing the suspended mass over the top of the box.

In order to improve the performance of this open-loop task given the uncertain length, the length estimate must be improved using the active estimation algorithm prior to attempting the task.



(a) Incorrect parameter plan failing to swing the mass into the box.



(b) Correct parameter plan successfully swinging the mass into the box.

Fig. 5. Baxter near the completion of the dynamic task for both incorrect and correct parameter plans.

B. Task Results with SAC Active Estimation

The second set of 9 trials were run with the first stage SAC estimator followed by the same task trajectory synthesis stage used in the previous set of trials. However, in this set of trials, the parameter value is identified using the estimator and the result is used in the task trajectory stage.

The resulting parameter values for the 9 trials throughout the active estimation stage are shown in Fig. 6. Estimation begins after the first second and the parameter values are updated as more information is acquired through the load cell measurements. Since the estimator uses a sufficient decrease condition on the least-squares problem, the value is only updated when the sufficient decrease is satisfied.

The estimates clearly converge toward the actual string length, which is noted by the dashed horizontal line. Qualitatively, the rate of convergence across all the trials is similar, with estimates beginning to converge around 2 to 4 seconds. This suggests that for this system, the trajectories generated by the SAC algorithm provide relatively similar levels of information despite the initial estimate. Since the estimator cost may not be convex, initial estimates too far from the actual value may not converge to the true value due to the presence of local minima. The mean estimate of ℓ from the 9 trials after 6 seconds was 0.367m with the actual string length set to 0.368m. The standard deviation of the final estimates is 0.0042m.

The generated and executed trajectories can be seen for one of the trials in Fig. 7. As discussed in Section III, the motion of the gripper is controlled to move along the Cartesian x-axis using PID control to follow the generated reference. The use of the Joint Trajectory Publisher along with the Baxter API results in reasonable tracking performance given the dynamic nature of the desired motion.

Immediately following the estimation stage for each trial, the task trajectory is synthesized and the task is attempted. The results of the task stage can be seen in the third column of Table I. Since the estimator has accurately identified the previously uncertain string length, the task is successfully completed from each initial length estimate for all 9 trials. For a more complete view of the experiment, the accompanying video shows one complete trial including the task trajectory optimization.

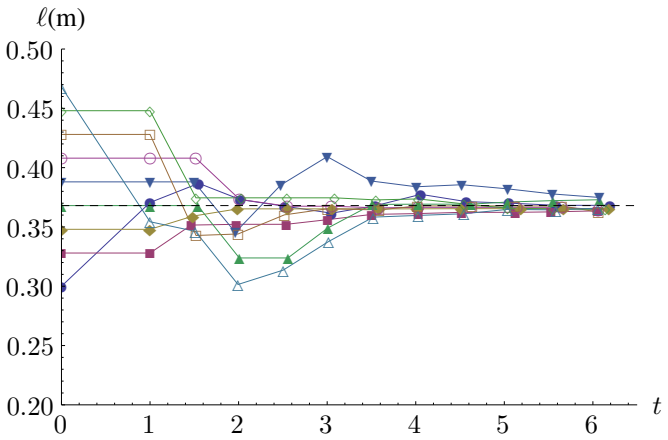


Fig. 6. Experimental trials on Baxter using different initial estimates of the string length. The dashed line indicates the actual measured length.

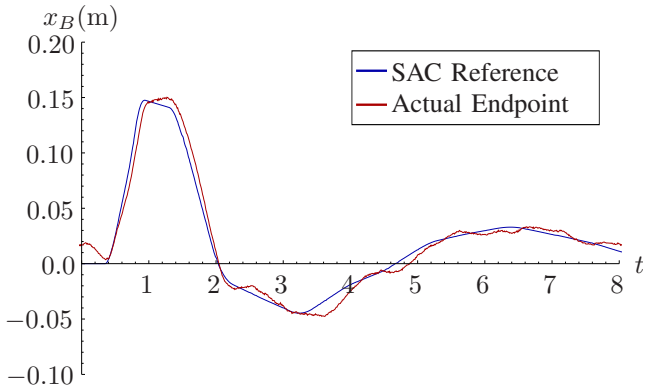


Fig. 7. Endpoint reference compared to actual endpoint position of the Baxter trial with $\ell_0 = 0.45$ m.

V. Conclusion

This paper presented an experimental implementation of control for real-time active estimation to improve open-loop task performance. Results from several trials on the Baxter robot with different initial estimates of the string length quickly converge to the actual length using a trajectory synthesized on-line using a Sequential Action Controller. This improvement in the parameter estimate allows a dynamic task to be completed despite a distribution of initial estimates of the parameter.

This work represents only one step toward improving robot learning on physical systems by exploiting dynamic models. One possible improvement may include the use of Lie groups as the fundamental tool to build the dynamic models. While the algorithms used to evaluate the Lie group models may be more complicated, they can facilitate better-posed solutions to the optimization problems, reducing singularities and angle wrapping issues common in robotic systems, especially non-holonomic systems.

Additionally, the extension to multiple parameters only requires that an optimality metric is set as shown in [11]; however, it would be useful for a system to realize which parameters need to be estimated in the first place. This could be achieved through forms of sensor fusion and covariance estimation or through exploration-based search algorithms. Eventually, the addition of a model creation and learning algorithm would enable a robot to develop not only estimates of the parameters, but also internal structure without prior knowledge of the internal dynamic model. The continued development of dynamics-based methods for robot learning will allow robots to learn and better interact with real-world objects and tasks in physical environments.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant CMMI 1334609. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] A. F. Emery and A. V. Nenarokomov, "Optimal experiment design," *Measurement Science and Technology*, vol. 9, pp. 864–876, June 1998.

[2] C. R. Rao and J. Wishart, "Minimum variance and the estimation of several parameters," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 43, p. 280, Oct. 1947.

[3] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation: Theory Algorithms and Software*. John Wiley & Sons, 2004.

[4] B. Armstrong, "On Finding Exciting Trajectories for Identification Experiments Involving Systems with Nonlinear Dynamics," *The International Journal of Robotics Research*, vol. 8, pp. 28–48, Dec. 1989.

[5] J. Swevers, C. Ganseman, D. Tukel, J. de Schutter, and H. Van Brussel, "Optimal robot excitation and identification," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 5, pp. 730–740, 1997.

[6] M. Gautier and W. Khalil, "Exciting Trajectories for the Identification of Base Inertial Parameters of Robots," *The International Journal of Robotics Research*, vol. 11, pp. 362–375, Aug. 1992.

[7] R. Mehra, "Optimal input signals for parameter estimation in dynamic systems–Survey and new results," *IEEE Transactions on Automatic Control*, vol. 19, pp. 753–768, Dec. 1974.

[8] T. L. Vincent, C. Novara, K. Hsu, and K. Poolla, "Input design for structured nonlinear system identification," *Automatica*, vol. 46, pp. 990–998, June 2010.

[9] G. Franceschini and S. Macchietto, "Model-based design of experiments for parameter precision: State of the art," *Chemical Engineering Science*, vol. 63, pp. 4846–4872, Oct. 2008.

[10] A. D. Wilson, J. A. Schultz, A. R. Ansari, and T. D. Murphey, "Real-time Trajectory Synthesis for Information Maximization using Sequential Action Control and Least-Squares Estimation," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4935–4940, 2015.

[11] A. D. Wilson, J. A. Schultz, and T. D. Murphey, "Trajectory Synthesis for Fisher Information Maximization," *IEEE Transactions on Robotics*, vol. 30, no. 6, pp. 1358–1370, 2014.

[12] A. Ansari and T. D. Murphey, "Sequential Action Control: Closed-Form Optimal Control for Nonlinear Systems," *IEEE Transactions on Robotics*.

[13] A. R. Ansari and T. D. Murphey, "Control-on-request: Short-burst assistive control for long time horizon improvement," in *2015 American Control Conf.*, pp. 1173–1180, July 2015.

[14] E. Johnson and T. Murphey, "Scalable Variational Integrators for Constrained Mechanical Systems in Generalized Coordinates," *IEEE Transactions on Robotics*, vol. 25, pp. 1249–1261, Dec. 2009.

[15] T. M. Caldwell, D. Coleman, and N. Correll, "Optimal Parameter Identification for Discrete Mechanical Systems with Application to Flexible Object Manipulation," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 898–905, 2014.

[16] R. Mao, Y. Yang, C. Fermuller, Y. Aloimonos, and J. S. Baras, "Learning hand movements from markerless demonstrations for humanoid tasks," in *2014 IEEE-RAS International Conference on Humanoid Robots*, pp. 938–943, Nov. 2014.

[17] C. Bowen and R. Alterovitz, "Closed-loop global motion planning for reactive execution of learned tasks," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1754–1760, 2014.

[18] E. Camacho and C. Alba, *Model Predictive Control*. Advanced Textbooks in Control and Signal Processing, Springer, 2013.

[19] L. Grüne and J. Pannek, *Nonlinear Model Predictive Control: Theory and Algorithms*. Communications and Control Engineering, Springer, 2011.

[20] J. Schultz, E. Johnson, and T. D. Murphey, "Trajectory optimization in discrete mechanics," in *Differential-Geometric Methods in Computational Multibody System Dynamics* (A. Müller and Z. Terze, eds.), Springer International Publishing, 2015 (expected). In Press.

[21] J. Hauser, "A projection operator approach to the optimization of trajectory functionals," in *World Congress*, vol. 15, p. 310, July 2002.

[22] Rethink Robotics Baxter Research Robot. [Online]. Available: http://www.rethinkrobotics.com/baxter-research-robot/.

[23] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, vol. 3, 2009.