Parallel dynamic programming for optimal experiment design in nonlinear systems

John Maidens, Andrew Packard, and Murat Arcak

***Abstract*— We present a method of computing optimal input trajectories for parameter estimation in nonlinear dynamical systems using dynamic programming. In contrast with previously published dynamic programming formulations, we avoid adding an equation for the dispersion to the system state, allowing for more efficient solutions. This method is applicable whenever the design metric is linear in the Fisher information and is applicable to a general class of noise models. We implement this algorithm in the Julia programming language, and exploit parallelism to increase computation speed. A motivating application for this investigation is the design of dynamic acquisition sequences for magnetic resonance imaging (MRI). We also benchmark the performance of our parallel implementation on a low-dimensional population dynamics model.**

I. Introduction

In this paper, we consider the problem of designing an experiment with the goal of estimating the parameters of a discrete-time nonlinear system from noisy observations of the system's output. The accuracy and reliability of parameter estimates in models of dynamical systems is dependent on the sequence of inputs used to drive the system, thus experimental design can be formulated as an optimal control problem using a scalar measure of the Fisher information as the objective function.

There is a significant body of literature on optimal experiment design for parameter estimation [1]–[4]. Historically, a great deal of this work has focused on linear dynamical systems and Gaussian noise models, where the input to the system is designed based on a frequency domain model with constraints on the input power spectrum. Here, we will approach this problem in the time domain, in order to be able to perform experiment design for systems with nonlinear dynamics. We show that for linear measures of the Fisher information matrix, the design objective can be expressed as a sum of stage costs for a new dynamic system augmented with sensitivity equations. Thus this problem has the optimal substructure required to allow us to formulate optimal experiment design as a dynamic programming problem [5], [6].

Dynamic programming formulations of optimal experiment design problems have been presented previously for aircraft model identification [7], [8]. We provide a broader formulation applicable to nonlinear dynamics and non-Gaussian noise models. We also avoid adding state equations corresponding to the dispersion matrix by limiting the class of metrics we consider to those that are linear in the Fisher information, increasing the scalability of the method.

In Section II we formulate the optimal experiment design problem and present the dynamic programming solution. In Section III we describe the problem that motivated this investigation: the design of optimized pulse sequences for dynamic MRI acquisitions. In Section IV we present a low-dimensional population dynamics model that shares some important features with the MRI model, which we will use to test and benchmark the implementation of our dynamic programming algorithm. Next, we discuss issues related to efficient parallelization of the algorithm in Section V. Finally, in Section VI we present a solution to the motivating MRI problem, achieved by running the dynamic programming algorithm in parallel on Amazon Web Services EC2 servers.

Software to implement the algorithms presented, and reproduce the figures from this paper is available at: `https://github.com/maidens/CDC-2016`.

II. Dynamic programming for optimal experiment design

We consider a discrete-time dynamical system with noisy observations

$$\begin{aligned} x_{t+1} &= f(x_t, u_t, \theta) \\ Y_t &\sim P_{x_t} \end{aligned} \tag{1}$$

where $x_t \in \mathbb{R}^n$ denotes the system's state, $u_t \in \mathbb{R}^m$ is a sequence of inputs to be designed and $\theta \in \mathbb{R}^p$ is a vector of unknown parameters that we wish to estimate. Observations $Y_t \in \mathbb{R}^p$ are drawn independently from a known distribution P_{x_t} that is parametrized by the system state x_t. We assume that for all $x_t \in \mathbb{R}^n$ the probability distribution P_{x_t} is absolutely continuous with respect to some measure μ and we denote its density with respect to μ by p_{x_t}. We further assume that this density is differentiable with respect to the parameter x_t and define the Fisher information about the state as

$$\mathcal{I}_{Y_t}(x_t) = \mathbb{E}_\theta \left[\left(\nabla_{x_t} \log p_{x_t}(Y_t) \right) \left(\nabla_{x_t} \log p_{x_t}(Y_t) \right)^T \right].$$

We consider this system over a finite horizon $0 \leq t \leq N$. Our goal is to design a sequence u that provides a maximal amount of information about the unknown parameter vector θ. Mathematically, we wish to choose u to maximize a

J. Maidens and M. Arcak are with the Department of Electrical Engineering & Computer Sciences, University of California, Berkeley, 253 Cory Hall, Berkeley, CA, 94720 USA e-mail: {maidens, arcak}@eecs.berkeley.edu. A. Packard is with the Department of Mechanical Engineering, University of California, Berkeley, 5116 Etcheverry Hall, Berkeley, CA, 94720 USA email: pack@me.berkeley.edu. Research supported in part by NSERC (Canada) postgraduate fellowship PGFD3-427610-2012, Amazon.com Inc. under and AWS Educate grant, and the National Science Foundation under grant ECCS-1405413.

function of the Fisher information that the joint output $Y = (Y_0, \ldots, Y_N)$ carries about the parameter θ. A function ϕ is chosen to be a measure of the "largeness" of the positive semidefinite matrix $\mathcal{I}$. Multiple choices for the function ϕ have been proposed [9]; here we use $\phi(\mathcal{I}) = \mathrm{tr}(\mathcal{I})$ (often called "T-optimal design"). In general this problem is nonconvex as a function of u. However, since the trace is linear, our objective function is additive and a global solution can be found using dynamic programming.

Note that since the Fisher information is a function of the true parameter value θ, the objective function for this optimal control problem depends on θ which is unknown *a priori*. There are several approaches to managing this uncertainty including: minimax optimal design where u is chosen to maximize the worst-case information among all values in some set Θ, Bayesian optimal design where u is chosen to maximize the expected information under some prior on θ, and choosing a nominal value θ_0 at which to maximize the information. For the sake of simplicity, in this paper we consider optimization about a nominal θ_0.

The following result allows us to compute the Fisher information about the parameter from the Fisher information about the state sequence. A proof of this proposition is given in the Appendix.

Proposition 1: Suppose that for all $x_t \in \mathbb{R}^n$ the density p_{x_t} is differentiable with respect to x_t and that there exists a μ-integrable function q with $\left|\frac{\partial p_{x_t}(y)}{\partial x_t}\right| \leq q(y)$ for all $y \in \mathbb{R}^p$. If f is C^1 in x_t and θ, then the Fisher information about the parameter θ can be computed as

$$\mathcal{I}_Y(\theta) = \sum_{t=0}^{N} (\nabla_\theta x_t)^T \mathcal{I}_{Y_t}(x_t)(\nabla_\theta x_t) \tag{2}$$

where $\nabla_\theta x_t$ denotes the Jacobian of x_t with respect to θ.

Thus, for the T-optimal design criterion, the objective function is given by

$$\mathrm{tr}(\mathcal{I}_Y(\theta)) = \sum_{t=0}^{N} \mathrm{tr}\left((\nabla_\theta x_t)^T \mathcal{I}_{Y_t}(x_t)(\nabla_\theta x_t)\right). \tag{3}$$

Applying the chain rule to (1), we get a dynamical system

$$\nabla_\theta x_{t+1} = \nabla_\theta f(x_t, u_t, \theta) + \nabla_x f(x_t, u_t, \theta)\ \nabla_\theta x_t \tag{4}$$

describing the time evolution of the sensitivities $\nabla_\theta x_t$. The dynamics (1) and (4) together with the cost function (3) define a discrete-time finite-horizon optimal control problem that can be solved via dynamic programming [5], [6]. In particular, we define the sequence of value functions J_k via

$$\begin{aligned} J_N(x_N, \nabla_\theta x_N) &= \mathrm{tr}\left((\nabla_\theta x_N)^T \mathcal{I}_{Y_N}(x_N)(\nabla_\theta x_N)\right) \\ J_k(x_k, \nabla_\theta x_k) &= \max_{u_k} \Big\{ \mathrm{tr}\left((\nabla_\theta x_k)^T \mathcal{I}_{Y_k}(x_k)(\nabla_\theta x_k)\right) \\ &\quad + J_{k+1}\Big(f(x_k, u_k, \theta), \nabla_\theta f(x_k, u_k, \theta) \\ &\quad + \nabla_x f(x_k, u_k, \theta)\ \nabla_\theta x_k\Big)\Big\}. \end{aligned} \tag{5}$$

If the control policy $u_k^* = \mu_k^*(x_k, \nabla_\theta x_k)$ maximizes the right hand side of (5) then u^* is globally optimal.

Related approaches to the optimal experiment design problem appear in [7] and [8] for continuous-time dynamical systems with Gaussian noise. However, a different objective function ϕ is used and these approaches require appending a nonlinear matrix differential equation for the dispersion (the inverse of the Fisher information) to the system state in addition to equation (4). By choosing the T-optimal design criterion $\phi(\mathcal{I}) = \mathrm{tr}(\mathcal{I})$, we are able to avoid adding an equation for the dispersion to the system state, allowing us to efficiently solve problems of larger dimension.

III. Motivating problem: Dynamic MRI acquisition

Magnetic resonance imaging (MRI) has traditionally focused on acquisition sequences that are static, in the sense that sequences typically wait for magnetization to return to equilibrium between acquisitions. Recently, researchers have demonstrated promising results based on dynamic acquisition sequences, in which spins are continuously excited by a sequence of random input pulses, without allowing the system to return to equilibrium between pulses. Model parameters corresponding to T_1 and T_2 relaxation, off-resonance and spin density are then estimated from the sequence of acquired data. This technique, termed magnetic resonance fingerprinting (MRF) has been shown to increase the sensitivity, specificity and speed of magnetic resonance studies [10], [11].

We believe that this technique could be further improved by replacing randomized input pulse sequences with sequences that have been optimized for informativeness about model parameters. To this end, we present a model of MR spin dynamics that describes the measured data as a function of T_1 and T_2 relaxation rates and the sequence of radio-frequency (RF) input pulses, used to excite the spins.

We model the spin dynamics via the equations (6) (displayed on the following page) where $x_{1,t}$ describes the longitudinal magnetization (parallel to the applied magnetic field), and $x_{2,t}$ describes the magnitude of the transverse magnetization (orthogonal to the applied magnetic field). Phase and off-resonance are neglected in this model to limit the state dimension. Control inputs u_t describe flip angles corresponding to RF excitation pulses that rotate the state about the origin. Between acquisitions, transverse magnetization decays according to the parameter $\theta_2 = e^{-\Delta t/T_2}$ and the longitudinal magnetization recovers to equilibrium (normalized such that the equilibrium is $x = [1 \quad 0]^T$) according to the parameter $\theta_1 = e^{-\Delta t/T_1}$.

We assume data is acquired immediately following the RF pulse, allowing us to make a noisy measurement of the transverse magnetization magnitude. The observed data Y_t are assumed to be Rician-distributed [12] with location parameter $[0 \quad 1]x_t$ and known scale parameter σ^2, described by the probability density given in (6). Here I_ν denotes the modified Bessel function of the first kind of order ν. The Fisher information about x_t corresponding to this distribution

$$x_{t+1} = f(x_t, u_t, \theta) = \begin{bmatrix} \cos u_t & -\sin u_t \\ \sin u_t & \cos u_t \end{bmatrix} \left(\begin{bmatrix} \theta_1 & 0 \\ 0 & \theta_2 \end{bmatrix} x_t + \begin{bmatrix} 1-\theta_1 \\ 0 \end{bmatrix} \right)$$
$$p_{x_t}(y) = \frac{y}{\sigma^2} \exp\left(-\frac{y^2 + x_{2,t}^2}{2\sigma^2} \right) I_0\left(\frac{y x_{2,t}}{\sigma^2} \right) \tag{6}$$

is given by

$$\mathcal{I}_{Y_t}(x_t) = \frac{1}{\sigma^2} \psi\left(\frac{[0 \quad 1] x_t}{\sigma} \right)$$

where $\psi(z) = -z^2 + \int_0^\infty y^3 \frac{I_1^2(yz)}{I_0(yz)} \exp\left(-\frac{1}{2}(y^2+z^2)\right) dy$ (see [13] for a derivation). This integral cannot be computed analytically, but is one-dimensional and parameter-independent, thus it can easily be evaluated numerically.

We see from the model (6) that magnetization in the transverse direction decays while magnetization in the longitudinal direction increases. However only the transverse component of the magnetization can be measured. Thus there is a trade-off between making measurements (which leads to loss of magnetization) and magnetization recovery. This is the trade-off that we hope to manage through optimal experiment design.

It should be noted that for system (6), the objective function $\mathrm{tr}(\mathcal{I}_Y(\theta))$ has many local optima as a function of the input sequence u. Thus, in contrast with [13], [14] which consider optimal experiment design for hyperpolarized MRI problems, for this model, local search methods provide little insight into what acquisition sequences are good. In contrast with the MRI model presented in [15], where global search heuristics are applied, in this model the dynamics are nonlinear with respect to the decision variables u_t. Therefore, we must attempt to optimize this objective function via the general dynamic programming formulation given in Section II. The results of this study are discussed in Section VI.

IV. Low-dimensional benchmark problem

In this section, we present a low-dimensional problem that is analogous to the MRI problem presented in Section III in the sense that it captures a trade-off between immediate and future signal strength than can be directly controlled via a sequence of inputs. We will use this problem in Section V to benchmark our dynamic programming implementation and to illustrate issues related to parallelization.

We consider a population of fruit flies, whose dynamics are assumed to evolve according to the discrete logistic equation

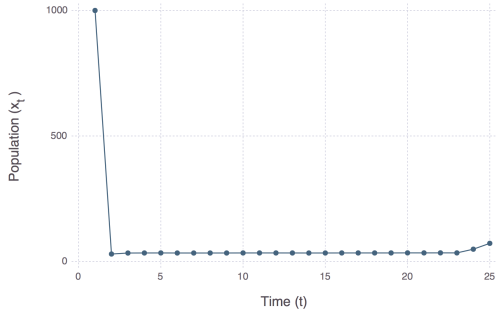
$$x_{t+1} = x_t + \theta x_t (K - x_t).$$

We want to design an experiment to estimate the reproduction rate θ, assuming a known carrying capacity K. To generate experimental data, we place a sequence of traps into the fly cage, each capturing a fraction u_t of the current fly population. By measuring the number of fruit flies caught in the trap, we wish to infer θ. The optimization problem thus consists of choosing the size of the traps (and hence the proportion of flies trapped) at each sampling interval.

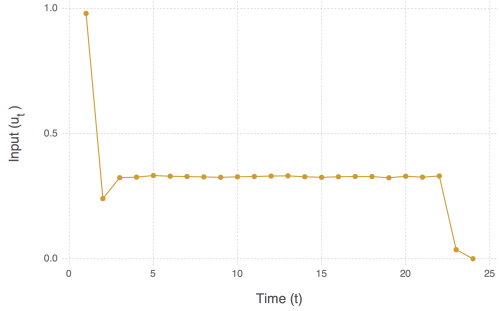
This leads to a model for the population dynamics together with the number of fruit flies trapped Y_t

$$\begin{aligned} x_0 &= K \\ x_{t+1} &= x_t(1-u_t) + \theta x_t(1-u_t)(K - x_t(1-u_t)) \\ Y_t &\sim Poisson(x_t u_t). \end{aligned} \tag{7}$$

For this problem, we approximate the functions J_k by evaluation on a state grid of size 50×50 and input grid of size 100. We assume that $K = 1000$ and optimize about the nominal parameter value $\theta_0 = 5 \times 10^{-4}$. The optimal inputs computed are plotted in Figure 1a. The corresponding state trajectory is shown in Figure 1b.



(a) Optimal input trajectory for (7) computed using dynamic programming



(b) State trajectory corresponding to the input given in Figure 1a

Fig. 1: Numerically-computed solution to the optimal experiment design problem defined by (7)

We see that the optimal observation scheme is to first capture a large fraction $u_1 = 0.9795$ of the flies, which drives the population to a low level thereby limiting the effect of growth saturation due to the carrying capacity. After this, we capture a fraction $u_t \approx 0.32$ of the flies, just enough to keep the population constant. This provides maximal sensitivity

to the growth rate θ in a neighbourhood of θ_0. Indeed, if $\theta > \theta_0$ we will see the the population of flies grow over time, whereas if $\theta < \theta_0$ the population will shrink.

V. Parallel Implementation

When the number of state variables and/or parameters is large, the number of points at which we must compute the value functions J_k becomes prohibitively large. However, the value of J_k at a particular grid point $(x, \nabla_\theta x)$ depends only on J_{k+1} and not the value of J_k elsewhere on the grid. Thus the evaluation of J_k on the grid can be performed entirely in parallel.

A. Low-dimensional benchmarks

In this section, we implement this parallelism via the map pattern [16], which applies the function J_k to all points on the state space grid. Distributing this task to multiple processors can then be handled automatically by the programming environment.

We present a comparison of implementations of dynamic programming for optimal experiment design. We use the algorithm to compute the value function sequence J_k for the population dynamics benchmark model given in Section IV. To compute J_k, we approximate the value function J_{k+1} from the previous iteration by linear interpolation between grid points. We compare the following four implementations:

- a serial implementation in the Julia language,
- a parallel implementation in the Julia language run with a single worker,
- a parallel implementation in the Julia language run with three workers (with shared memory on a single multicore machine).
- a parallel implementation in the Julia language run with 31 workers (with shared memory on a single multicore machine, when applicable).

Simulations are performed on

- a Macbook laptop (2.3 GHz quad-core Intel Core i7 Ivy Bridge processor, 8GB memory) and
- an Amazon Web Services (AWS) Elastic Compute Cloud (EC2) `cc2.8xlarge` instance (2.59 GHz 32 vCPU machine with two Intel Xeon E5-2670 Processors, 60GB memory).

Code to run the simulations is available at `https://github.com/maidens/CDC-2016`. A comparison of the running times are given in Table I for a problem solved on a (state 1) $\times$ (state 2) $\times$ (input) grid of 50 $\times$ 50 $\times$ 100 = 250,000 points and in Table II on a grid of 5 $\times$ 5 $\times$ 10000 = 250,00 points.

The results for the 50 $\times$ 50 $\times$ 100 grid are unexpected as the serial implementation in Julia runs faster than any of the parallel implementations. This is not the case for the same problem solved on a 5 $\times$ 5 $\times$ 10000 grid which contains the same total number of grid points, but distributed differently between the state and input grids. We see that the two grids lead to similar running times when run in serial, but very different running times when run in parallel. We will explore the source of this discrepancy in the next section.

B. Parallelism and data blocking

The Julia programming language implements parallelism via one-sided message passing based on remote references and remote calls [17]. The architecture of a simple parallel program running on a machine with $n+1$ cores is shown in Fig. 2. One processor core acts as a coordinator in charge of assigning remote function calls to be evaluated by each of n worker processor cores.

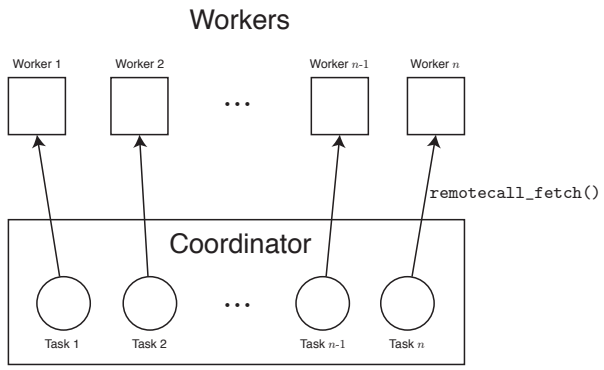


Fig. 2: Implementation architecture of the parallel map function in Julia

If the amount of time that a worker requires to evaluate the remote call is large (*i.e.* greater than by a factor of n) compared with the time it takes for the coordinator to manage that function call, then we expect to be able to achieve nearly a factor of n speedup in computation. However, if each function call can be evaluated relatively quickly, then a coordination bottleneck can occur, limiting the speedup that can be achieved via parallelization.

We can overcome this limitation by grouping data such that each worker is assigned an entire block of data to process each time that a remote call is initiated. In Fig. 3 we plot the running time of the dynamic programming algorithm as a function of data block size. We see that the size of the blocks has a very significant effect on the algorithm's speed. When the group size is small, each function call is very fast and a coordination bottleneck occurs, slowing the algorithm significantly. When the group size is too large then some CPUs sit idle, also slowing the algorithm. But by choosing an appropriate group size, a speedup on the order of $2^4\times$ can be achieved.

VI. Solution to the Motivating MRI Problem

In this section, we present a solution to the motivating problem introduced in Section III. We consider the case of designing a flip angle sequence that is maximally informative about the T_1 relaxation rate, assuming that T_2 is known.

The value function and corresponding optimal control input are computed using a state grid of size 20 $\times$ 20 $\times$ 30 $\times$ 30 and input grid of size 62 ($-\pi \leq u_t \leq \pi$ with $\Delta u_t = 0.1$), with parallelization implemented using a synchronous parallel for loop, taking care to block data appropriately to ensure parallel speedup. The computation is

		Julia serial	Julia parallel – 1 worker	Julia parallel – 3 workers	Julia parallel – 31 workers
Macbook (4 vCPU)	time	8.22 s	40.1 s	15.3 s	Not
	memory	2.362 GB	4.263 GB	4.266 GB	Applicable
EC2 (32 vCPU)	time	7.41 s	37.7 s	14.0 s	13.8 s
	memory	2.361 GB	3.707 GB	3.683 GB	3.690 GB

TABLE I: Run time comparison of three implementations of dynamic programming for optimal experiment design computed on a state grid of 50×50 points and an input grid of 100 points.

		Julia serial	Julia parallel – 1 worker	Julia parallel – 3 workers	Julia parallel – 31 workers
Macbook (4 vCPU)	time	7.33 s	9.13 s	3.06 s	Not
	memory	2.308 GB	58.308 MB	31.833 MB	Applicable
EC2 (32 vCPU)	time	7.38 s	8.66 s	2.84 s	1.29 s
	memory	2.322 GB	46.887 MB	25.960 MB	29.208 MB

TABLE II: Run time comparison of three implementations of dynamic programming for optimal experiment design computed on a state grid of 5×5 points and an input grid of 10000 points.

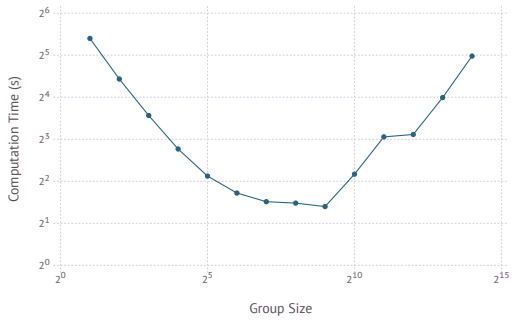


Fig. 3: Comparison of running time as a function of group size for dynamic programming for optimal experiment design computed on a state grid of 128×128 points and an input grid of 100 points. Computations are performed on an Amazon EC2 `c4.8xlarge` instance (with 36 vCPUs).

performed on an 32 vCPU `cc2.8xlarge` EC2 instance, which for a horizon of $N = 30$ samples has a runtime of 683.3 seconds. The optimal flip angle sequence resulting for the dynamic programming algorithm, assuming that this system is initialized at the equilibrium $x_0 = [1 \quad 0]^T$, $\nabla_{\theta_1} x_0 = [0 \quad 0]^T$, is shown in Fig. 4. The corresponding state trajectory is plotted in Fig. 5.

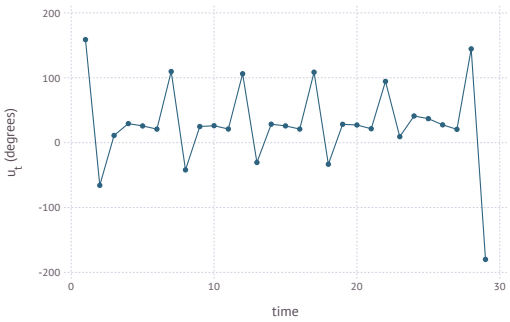


Fig. 4: Optimized flip angle input sequence

We see that the maximally informative control law converges to a repeating sequence of period 5. This is an

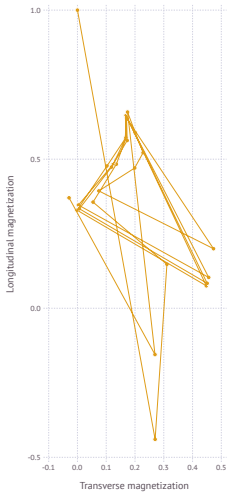


Fig. 5: State trajectories corresponding to optimized flip angle sequence. The state is initialized at the equilibrium $x_0 = [1 \quad 0]$ and is driven by the control input to a periodic trajectory.

interesting behavior that warrants further study. It is possible that such periodic pulse sequences could be used to develop highly-sensitive dynamic MRI acquisition methods.

To confirm that the optimized pulse sequence leads to more accurate parameter estimates than a random pulse sequence, we compare the result of maximum likelihood estimates of θ_1 between the two pulse sequences. This comparison is shown in Fig. 6 for a sample of 1000 estimation experiments. We see that the estimates resulting from the optimized input sequence have lower bias and lower variance than those corresponding to the random sequence.

VII. Conclusion

We have shown that dynamic programming can be used to design globally optimal input trajectories for estimating parameters in nonlinear dynamical systems with non-Gaussian noise models. The execution speed of the algorithm can be increased through parallelization, though it is necessary to

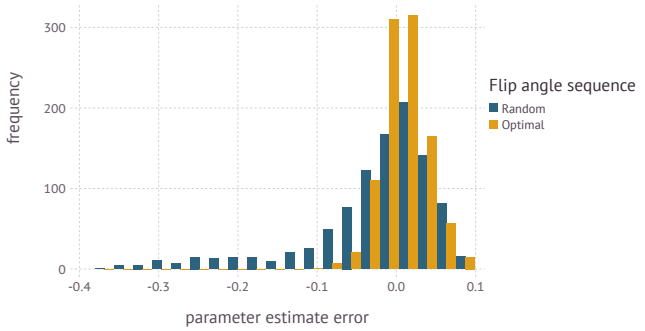


Fig. 6: Histogram of θ_1 parameter estimates compared between a random input and a sequence optimized using our dynamic programming algorithm. Our algorithm leads to more accurate estimates of the model parameter.

carefully coordinate parallel processes to prevent bottlenecks. This algorithm has enabled us to compute an optimized pulse sequence for a dynamic MRI acquisition problem that has so far been intractable using existing methods.

To enable these results to scale to higher-dimensional systems, in the future we would like to explore more efficient models for interpolating the value function such as neural networks [18] and Kriging models [19], [20].

APPENDIX
PROOF OF PROPOSITION 1

Proof: First, note that the hypotheses of this proposition provide sufficient regularity to exchange the order of differentiation with respect to θ_i and integration with respect to y_t. Therefore for all $i = 1, \dots, p$ and $t = 0, \dots, N$

$$\begin{aligned}\mathbb{E}_\theta\left[\frac{\partial \log p_\theta(Y_t)}{\partial \theta_i}\right] &= \int \frac{\partial \log p_\theta(y_t)}{\partial \theta_i} p_\theta(y_t) d\mu(y_t) \\ &= \int \frac{\partial p_\theta(y_t)}{\partial \theta_i} d\mu(y_t) \\ &= \frac{\partial}{\partial \theta_i} \int p_\theta(y_t) d\mu(y_t) \\ &= \frac{\partial}{\partial \theta_i} 1 = 0.\end{aligned}$$

Now for all $i, j = 1, \dots, p$ we can compute the (i,j)-th entry of $\mathcal{I}_\theta(Y)$ as

$$\begin{aligned}\mathcal{I}_Y(\theta)_{i,j} &= \mathbb{E}_\theta\left[\frac{\partial \log p_\theta(Y)}{\partial \theta_i}\frac{\partial \log p_\theta(Y)}{\partial \theta_j}\right] \\ &= \mathbb{E}_\theta\left[\left(\sum_{t=0}^N \frac{\partial \log p_\theta(Y_t)}{\partial \theta_i}\right)\left(\sum_{s=0}^N \frac{\partial \log p_\theta(Y_s)}{\partial \theta_j}\right)\right] \\ &= \sum_{t=0}^N \mathbb{E}_\theta\left[\frac{\partial \log p_\theta(Y_t)}{\partial \theta_i}\frac{\partial \log p_\theta(Y_t)}{\partial \theta_j}\right] \\ &\quad + \sum_{t=0}^N \sum_{s=0, s\neq t}^N \mathbb{E}_\theta\left[\frac{\partial \log p_\theta(Y_t)}{\partial \theta_i}\right]\mathbb{E}_\theta\left[\frac{\partial \log p_\theta(Y_s)}{\partial \theta_j}\right] \\ &= \sum_{t=0}^N \mathbb{E}_\theta\left[\frac{\partial \log p_{x_t(\theta)}(Y_t)}{\partial \theta_i}\frac{\partial \log p_{x_t(\theta)}(Y_t)}{\partial \theta_j}\right] \\ &= \sum_{t=0}^N \mathbb{E}_\theta\left[\langle \frac{\partial x_t}{\partial \theta_i}, \nabla_{x_t} \log p_{x_t}(Y_t)\rangle \langle \frac{\partial x_t}{\partial \theta_j}, \nabla_{x_t} \log p_{x_t}(Y_t)\rangle\right] \\ &= \sum_{t=0}^N \frac{\partial x_t}{\partial \theta_i}^T \mathbb{E}_\theta\left[\left(\nabla_{x_t} \log p_{x_t}(Y_t)\right)\left(\nabla_{x_t} \log p_{x_t}(Y_t)\right)^T\right]\frac{\partial x_t}{\partial \theta_y} \\ &= \sum_{t=0}^N \frac{\partial x_t}{\partial \theta_i}^T \mathcal{I}_{Y_t}(x_t)\frac{\partial x_t}{\partial \theta_y}.\end{aligned}$$

So

$$\mathcal{I}_Y(\theta) = \sum_{t=0}^N (\nabla_\theta x_t)^T \mathcal{I}_{Y_t}(x_t)(\nabla_\theta x_t).$$

■

REFERENCES

[1] M. Gevers, X. Bombois, R. Hildebrand, and G. Solari, "Optimal experiment design for open and closed-loop system identification," *Communications in Information and Systems*, vol. 11, no. 3, pp. 197–224, 2011.

[2] G. Goodwin and R. Payne, *Dynamic System Identification: Experiment Design and Data Analysis*. Academic Press, 1977.

[3] L. Ljung, *System Identification: Theory for the User*. Pearson Education, 1999.

[4] É. Walter and L. Pronzato, *Identification of parametric models from experimental data*, ser. Communications and control engineering. Springer, 1997.

[5] R. Bellman, *Dynamic Programming*. Princeton University Press, 1957.

[6] D. P. Bertsekas, *Dynamic Programming and Optimal Control, Volume 1*. Athena Scientific, 1995.

[7] R. T. N. Chen, "Input design for aircraft parameter identification: Using time-optimal control formulation," in *Methods for Aircraft State and Parameter Identification, Advisory Group for Aerospace Research and Development (AGARD), Conference Proceedings no. 172*, 1975.

[8] E. A. Morelli and V. Klein, "Optimal input design for aircraft parameter estimation using dynamic programming principles," in *AIAA Atmospheric Flight Mechanics Conference paper 90-2801*, 1990.

[9] F. Pukelsheim, *Optimal Design of Experiments*. Society for Industrial and Applied Mathematics, 2006.

[10] D. Ma, V. Gulani, N. Seiberlich, K. Liu, J. L. Sunshine, J. L. Duerk, and M. A. Griswold, "Magnetic resonance fingerprinting," *Nature*, vol. 495, no. 7440, pp. 187–192, 03 2013.

[11] M. Davies, G. Puy, P. Vandergheynst, and Y. Wiaux, "A compressed sensing framework for magnetic resonance fingerprinting," *SIAM Journal on Imaging Sciences*, vol. 7, no. 4, pp. 2623–2656, 2014.

[12] H. Gudbjartsson and S. Patz, "The Rician distribution of noisy MRI data," *Magnetic Resonance in Medicine*, vol. 34, no. 6, pp. 910–914, 1995.

[13] J. Maidens, P. E. Z. Larson, and M. Arcak, "Optimal experiment design for physiological parameter estimation using hyperpolarized carbon-13 magnetic resonance imaging," in *Proceedings of the American Control Conference (ACC)*, 2015, pp. 5770–5775.

[14] J. Maidens, J. W. Gordon, M. Arcak, and P. E. Z. Larson, "Optimizing flip angles for metabolic rate estimation in hyperpolarized carbon-13 MRI," *IEEE Transactions on Medical Imaging*, 2016, to appear.

[15] J. Maidens and M. Arcak, "Semidefinite relaxations in optimal experiment design with application to substrate injection for hyperpolarized MRI," in *Proceedings of the American Control Conference (ACC)*, 2016, to appear.

[16] M. McCool, J. Reinders, and A. Robison, *Structured Parallel Programming: Patterns for Efficient Computation*. Morgan Kaufmann, 2012.

[17] Julia Language Documentation, "Parallel computing," http://docs.julialang.org/en/release-0.4/manual/parallel-computing/, accessed: 2015-03-11.

[18] D. P. Bertsekas and J. Tsitsiklis., *Neuro-Dynamic Programming*. Athena Scientific, 1995.

[19] D. Krige, "A statistical approach to some basic mine valuation problems on the Witwatersrand," *Journal of the Chemical, Metallurgical and Mining Society of South Africa*, no. 52, 1951.

[20] N. Cressie, *Statistics for spatial data*, ser. Wiley series in probability and mathematical statistics: Applied probability and statistics. J. Wiley, 1993.