Dynamic Parametric Sensitivity Optimization Using Simultaneous Discretization in JModelica.org

Magnusson, Fredrik; Palmer, Kyle; Han, Lu; Bollas, George

Published in:
2015 International Conference on Complex Systems Engineering

DOI:
10.1109/ComplexSys.2015.7385980

2015

Document Version:
Peer reviewed version (aka post-print)

Dynamic Parametric Sensitivity Optimization Using Simultaneous Discretization in JModelica.org

Fredrik Magnusson*, Kyle Palmer†, Lu Han†, George Bollas†
*Department of Automatic Control
Lund University, SE-221 00 Lund, Sweden
Email: fredrik.magnusson@control.lth.se
†Department of Chemical & Biomolecular Engineering
University of Connecticut, Storrs, 191 Auditorium Road, Unit 3222, Storrs, CT, 06269-3222, USA
Email: bollas@engr.uconn.edu

***Abstract*—Dynamic optimization problems involving parametric sensitivities, such as optimal experimental design, are typically solved using shooting-based methods, while leveraging numerical integrators with sensitivity computation capabilities. In this paper we present how simultaneous discretization can be employed to solve these problems, by augmenting the dynamic optimization problems with forward sensitivity equations. We present an implementation of this approach in the open-source, Modelica-based tool JModelica.org, which addresses the need for solving optimal experimental design problems in Modelica tools. The implementation is demonstrated on a fed-batch reactor and a plate-fin heat exchanger.**

I. Introduction

Dynamic optimization problems (DOP) occur in many different fields and contexts, including optimal control, parameter and state estimation, and design optimization. Another class of DOPs is Optimal Experimental Design (OED) [1]–[3], which is a model-based methodology for maximizing the useful information that can be extracted from any experiment. Experimental data are required to assess the applicability of the model and to estimate the empirical parameters in order to arrive at a model that is accurate and reliable. In a model-based OED approach, the mathematical model is used to solve a dynamic optimization problem for the experimental design. This paper is focused on model-based OED for parameter estimation, wherein the objective function is composed of the sensitivities of the measurements with respect to unknown parameters.

There are many approaches to numerically solving DOPs [4], [5], which all involve the discretization of differential equations in order to obtain a nonlinear program (NLP). Most of these—such as control vector parametrization and direct multiple shooting—utilize embedded numerical integrators to treat the differential equations. The computation of the parametric sensitivities in the optimization formulation is then straightforward by relying on solvers with sensitivity calculation capabilities. These are also the methods that are typically used for OED [6]–[8], in order to reduce the size of the resulting NLP. In this paper we explore the use of simultaneous discretization approaches for DOPs involving parametric sensitivities. These approaches fully discretize the state and control variables and encode the discretized equations in the NLP. This eliminates the need for an embedded numerical integrator, but on the other hand results in a very large NLP. Also, for DOPs that involve parametric sensitivities, numerical integrators can no longer be utilized for sensitivity computations. We treat this by essentially transforming the DOP that involves parametric sensitivities into one that does not, by transforming the parametric sensitivities into state variables using standard sensitivity analysis techniques. The main advantages of simultaneous approaches is that while the resulting NLP is large, it is also sparse and has less severe nonlinearities. They also allow for more efficient treatment of path constraints.

Modelica [9] is an object-oriented language for the modeling of heterogeneous physical systems. It is based on a declarative equation-based paradigm designed for both textual and graphical modeling, whose underlying mathematical formalism is that of DAE systems. JModelica.org [10] is a Modelica tool for model-based analysis of large-scale dynamic systems. The support for solving OED problems is close to nonexistent in today's Modelica tools, which this paper addresses.

The contribution of this paper is twofold. The first is the exploration of employing simultaneous, rather than sequential, discretization methods for the numerical solution of OED problems. The second contribution is an open-source toolchain for OED using Modelica. Recent and related work in these directions is [11] and [12].

The outline of the paper is as follows. In Section II we present the theoretical foundations of the proposed framework and also describe the relevant modules of JModelica.org. In Section III we discuss how the theory is used to realize a framework for solving DOPs involving parametric sensitivities—in particular OED—and its implementation in JModelica.org. In Section IV we demonstrate the framework on two OED examples and compare the performance with other frameworks. In Section V we discuss the framework properties and possible extensions.

II. Background

In this section we present the theoretical background of dynamic optimization, OED, and sensitivity analysis, which is needed to realize the proposed framework.

A. Dynamic Optimization

The mathematical foundation of Modelica is that of implicit differential-algebraic equations (DAE) in the form

$$\boldsymbol{F}(\dot{\boldsymbol{x}}(t), \boldsymbol{x}(t), \boldsymbol{w}(t), \boldsymbol{u}(t), \boldsymbol{p}, \boldsymbol{\theta}) = \boldsymbol{0}, \tag{1}$$

where $t \in [t_0, t_f]$ is the time, $\boldsymbol{x}$ is the differential variables, $\boldsymbol{w}$ is the algebraic variables, $\boldsymbol{u}$ is the system inputs, $\boldsymbol{p}$ is the free parameters, and $\boldsymbol{\theta}$ is the parameters for which sensitivities are needed. Note that a model parameter can be included in both $\boldsymbol{p}$ and $\boldsymbol{\theta}$. In this work we restrict ourselves to DAEs whose residual function $\boldsymbol{F}$ is sufficiently smooth, which in particular excludes hybrid DAEs. We also assume the DAE to be of most index one, which however is without great loss of generality, as the JModelica.org compiler will transform the DAE into index one using the method of dummy derivatives [13]. The differential variable $\boldsymbol{x}$ thus corresponds to the system state. We will treat initial conditions on the general implicit form

$$\boldsymbol{F}_0(\dot{\boldsymbol{x}}(t_0), \boldsymbol{x}(t_0), \boldsymbol{w}(t_0), \boldsymbol{u}(t_0), \boldsymbol{p}, \boldsymbol{\theta}) = \boldsymbol{0}, \tag{2}$$

although they are most commonly available on the explicit form

$$\boldsymbol{x}(t_0) = \boldsymbol{x}_0(\boldsymbol{p}, \boldsymbol{\theta}), \tag{3}$$

possibly depending on parameters.

A general formulation of a DOP is

$$\begin{aligned}
\text{min.} \quad & f(\boldsymbol{x}(t_1), \boldsymbol{w}(t_1), \ldots, \boldsymbol{x}(t_{n_m}), \boldsymbol{w}(t_{n_m})), && \text{(4a)}\\
\text{w.r.t.} \quad & \boldsymbol{x} : [t_0, t_f] \to \mathbb{R}^{n_x}, \quad \boldsymbol{w} : [t_0, t_f] \to \mathbb{R}^{n_w},\\
& \boldsymbol{u} : [t_0, t_f] \to \mathbb{R}^{n_u}, \quad \boldsymbol{p} \in \mathbb{R}^{n_p},\\
\text{s.t.} \quad & \boldsymbol{F}(\dot{\boldsymbol{x}}(t), \boldsymbol{x}(t), \boldsymbol{w}(t), \boldsymbol{u}(t), \boldsymbol{p}, \boldsymbol{\theta}) = \boldsymbol{0}, && \text{(4b)}\\
& \boldsymbol{F}_0(\dot{\boldsymbol{x}}(t_0), \boldsymbol{x}(t_0), \boldsymbol{w}(t_0), \boldsymbol{u}(t_0), \boldsymbol{p}, \boldsymbol{\theta}) = \boldsymbol{0}, && \text{(4c)}\\
& \boldsymbol{g}(\dot{\boldsymbol{x}}(t), \boldsymbol{x}(t), \boldsymbol{w}(t), \boldsymbol{u}(t), \boldsymbol{p}, \boldsymbol{\theta}) \leq \boldsymbol{0}, && \text{(4d)}\\
& \forall t \in [t_0, t_f],
\end{aligned}$$

where (4a) is the objective, (4b) are the differential equations governing the system dynamics and (4c) are the corresponding initial conditions, and (4d) are general inequality constraints, such as lower and upper bounds on state or input variables. A more typical objective function than (4a) is a Bolza functional [14, Section 3.3]. However, (4a), which can be seen as a generalized Mayer term [15], has been chosen due to its compatibility with discrete measurements as is typically the case in OED, where f is a scalar-valued function depending on system variable values at discrete time points—typically measurement points—$t_i \in [t_0, t_f]$ and n_m is the number of such points.

B. Optimal Experiment Design

The purpose of OED is to maximize the information content of a set of experiments to facilitate parameter estimation. In general, the experiment design space consists of the system inputs $\boldsymbol{u}$, initial state $\boldsymbol{x}_0$, measurement time points t_i, and experiment duration t_f, giving us the general design vector

$$\boldsymbol{\phi}' = [\boldsymbol{u}, \boldsymbol{x}_0, t_i, t_f]. \tag{5}$$

Assuming state- and parameter-independent, zero-mean Gaussian noise, the information is measured by the Fisher information matrix

$$\mathbf{H}_{\boldsymbol{\theta}} = \sum_{i=1}^{n_m} \sum_{j=1}^{n_m} \sigma_{i,j} \mathbf{Q}_i^T \mathbf{Q}_j, \tag{6}$$

where σ is the experimental variance matrix and

$$\mathbf{Q}_i = \begin{pmatrix} \nabla_{\boldsymbol{\theta}} y_i(t_1) \\ \nabla_{\boldsymbol{\theta}} y_i(t_2) \\ \vdots \\ \nabla_{\boldsymbol{\theta}} y_i(t_{n_m}) \end{pmatrix} \tag{7}$$

contains the parametric sensitivities of measured variable y_i, with respect to all uncertain parameters $\boldsymbol{\theta}$, at all measurement points t_i. In this paper we examine the case where only a single experiment is performed and the measurement time points t_i and the duration t_f are fixed a priori, giving us the simplified design vector

$$\boldsymbol{\phi} = [\boldsymbol{u}, \boldsymbol{x}_0]. \tag{8}$$

Maximizing the Fisher information matrix is equivalent to minimizing its inverse, the variance-covariance matrix. By doing so, the inference regions of the model parameters are decreased and the parameter precision is improved. The objective of OED is thus to maximize some scalar metric of $\mathbf{H}_{\boldsymbol{\theta}}$ or to minimize some scalar metric of $\mathbf{H}_{\boldsymbol{\theta}}^{-1}$. Three common measures of the information are the D-criterion, which maximizes the determinant of the $\mathbf{H}_{\boldsymbol{\theta}}$, A-criterion, which minimizes the trace of $\mathbf{H}_{\boldsymbol{\theta}}^{-1}$, and E-criterion, which maximizes the minimum eigenvalue of $\mathbf{H}_{\boldsymbol{\theta}}$. We will focus on A-optimal design for simplicity, but the method is extensible to other designs. The optimal design in this work is thus

$$\begin{aligned}
\boldsymbol{\phi}_A = \arg\min_{\boldsymbol{\phi}} \quad & \operatorname{tr}\left(\mathbf{H}_{\boldsymbol{\theta}}^{-1}\right), && \text{(9a)}\\
\text{subject to} \quad & \boldsymbol{F}(\dot{\boldsymbol{x}}(t), \boldsymbol{x}(t), \boldsymbol{w}(t), \boldsymbol{u}(t), \boldsymbol{\theta}) = \boldsymbol{0}, && \text{(9b)}\\
& \boldsymbol{x}(t_0) = \boldsymbol{x}_0, && \text{(9c)}\\
& \boldsymbol{g}(\dot{\boldsymbol{x}}(t), \boldsymbol{x}(t), \boldsymbol{w}(t), \boldsymbol{u}(t)) \leq \boldsymbol{0}, && \text{(9d)}\\
& \forall t \in [t_0, t_f],
\end{aligned}$$

C. Parametric Sensitivity Analysis

A common approach for computing parametric sensitivities for explicit ordinary differential equations is to apply the chain rule of differentiation to the differential equations [16]. This way of computing forward sensitivities can also be applied to the implicit DAE system (1) and the initial equations (2), yielding

$$\begin{aligned}
&\nabla_{\dot{\boldsymbol{x}}} \boldsymbol{F}(\dot{\boldsymbol{x}}(t), \boldsymbol{x}(t), \boldsymbol{w}(t), \boldsymbol{u}(t), \boldsymbol{p}, \boldsymbol{\theta}) \dot{\boldsymbol{s}}(t)\\
&+\nabla_{\boldsymbol{x}} \boldsymbol{F}(\dot{\boldsymbol{x}}(t), \boldsymbol{x}(t), \boldsymbol{w}(t), \boldsymbol{u}(t), \boldsymbol{p}, \boldsymbol{\theta}) \boldsymbol{s}(t)\\
&+\nabla_{\boldsymbol{w}} \boldsymbol{F}(\dot{\boldsymbol{x}}(t), \boldsymbol{x}(t), \boldsymbol{w}(t), \boldsymbol{u}(t), \boldsymbol{p}, \boldsymbol{\theta}) \boldsymbol{z}(t)\\
&+\nabla_{\boldsymbol{\theta}} \boldsymbol{F}(\dot{\boldsymbol{x}}(t), \boldsymbol{x}(t), \boldsymbol{w}(t), \boldsymbol{u}(t), \boldsymbol{p}, \boldsymbol{\theta}) \mathbf{1} = \boldsymbol{0},
\end{aligned} \tag{10a}$$

$$\begin{aligned}
&\nabla_{\dot{\boldsymbol{x}}} \boldsymbol{F}_0(\dot{\boldsymbol{x}}(t_0), \boldsymbol{x}(t_0), \boldsymbol{w}(t_0), \boldsymbol{u}(t_0), \boldsymbol{p}, \boldsymbol{\theta}) \dot{\boldsymbol{s}}(t_0)\\
&+\nabla_{\boldsymbol{x}} \boldsymbol{F}_0(\dot{\boldsymbol{x}}(t_0), \boldsymbol{x}(t_0), \boldsymbol{w}(t_0), \boldsymbol{u}(t_0), \boldsymbol{p}, \boldsymbol{\theta}) \boldsymbol{s}(t_0)\\
&+\nabla_{\boldsymbol{w}} \boldsymbol{F}_0(\dot{\boldsymbol{x}}(t_0), \boldsymbol{x}(t_0), \boldsymbol{w}(t_0), \boldsymbol{u}(t_0), \boldsymbol{p}, \boldsymbol{\theta}) \boldsymbol{z}(t_0)\\
&+\nabla_{\boldsymbol{\theta}} \boldsymbol{F}_0(\dot{\boldsymbol{x}}(t_0), \boldsymbol{x}(t_0), \boldsymbol{w}(t_0), \boldsymbol{u}(t_0), \boldsymbol{p}, \boldsymbol{\theta}) \mathbf{1} = \boldsymbol{0},
\end{aligned} \tag{10b}$$

where $\dot{\boldsymbol{s}} = \nabla_{\boldsymbol{\theta}} \dot{\boldsymbol{x}}$, $\boldsymbol{s} = \nabla_{\boldsymbol{\theta}} \boldsymbol{x}$, and $\boldsymbol{z} = \nabla_{\boldsymbol{\theta}} \boldsymbol{w}$.

This approach gives rise to $(n_x + n_w) \cdot n_\theta$ additional implicit DAEs, which are added to the original DAE system (1). Numerical integration of the augmented system then computes the parametric sensitivities, with a clear computational burden for systems with many equations and parameters for which to compute sensitivities for.

D. JModelica.org Toolchain

An overview of the JModelica.org toolchain for optimization and simulation used in this paper is shown in Figure 1. It uses the modeling language Modelica [9] to describe system dynamics, and the optimization problem is formulated with the use of the Modelica language extension Optimica [17]. The user interacts with the different modules of JModelica.org via Python. One such module is a high-level and efficient framework for dynamic optimization, in which the Modelica and Optimica code is compiled and transferred into CasADi Interface [18]. This procedure creates a symbolic representation of the optimization problem and the model equations. CasADi Interface serves as a three-way interface between the JModelica.org compiler, the numerical dynamic optimization algorithms, and the user. The main optimization algorithm in JModelica.org is based on direct local collocation [4], [15], which simultaneously discretizes the infinite-dimensional DOP into a finite-dimensional NLP. A local solution to the NLP is then computed by IPOPT [19], utilizing first- and second-order derivatives computed by algorithmic differentiation using CasADi [20].

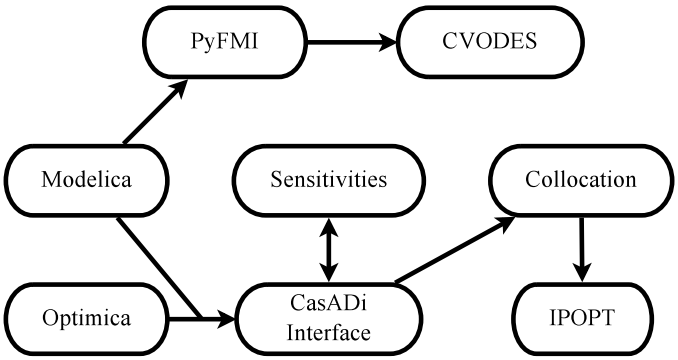

Figure 1. The JModelica.org toolchain for simulation and optimization. For simulation purposes, C code is generated based on the Modelica code and then connected to CVODES via PyFMI. For optimization purposes, the Modelica and Optimica code is symbolically transferred to CasADi Interface. The optimization problem is then solved using direct collocation and IPOPT. The Sensitivities module is described in Section III.

JModelica.org's simulation framework is based on the Functional Mockup Interface (FMI) [21]. After the system has been transformed into an ODE and exported via FMI, it is imported again via PyFMI and connected to ODE solvers via Assimulo [22], in particular CVODES [16] from the SUNDIALS suite, which supports the computation of parametric sensitivities.

III. Framework Implementation

The dynamic optimization framework in JModelica.org can solve problems in the form of (4) (with various generalizations not considered in this paper). However, the objective function (9a) does not fit into the form of (4), due to the dependence on parametric sensitivities. The proposed way of handling this is to augment (4) with state and algebraic variables corresponding to the parametric sensitivities s and z and the corresponding forward sensitivity equations and initial conditions, as described in Section II-C. The parametric sensitivities are then available as state or algebraic variables, and the objective (9a) becomes a special case of (4a). The transformed formulation of the A-optimal design problem is thus

$$\begin{aligned}
\text{min.} \quad & \operatorname{tr}\left(\left(\sum_{i=1}^{n_m}\sum_{j=1}^{n_m}\sigma_{i,j}\mathbf{Q}_i^T\mathbf{Q}_j\right)^{-1}\right), && \text{(11a)}\\
\text{w.r.t.} \quad & \boldsymbol{x}:[t_0,t_f]\to\mathbb{R}^{n_x}, \quad \boldsymbol{w}:[t_0,t_f]\to\mathbb{R}^{n_w},\\
& \boldsymbol{s}:[t_0,t_f]\to\mathbb{R}^{n_x\cdot n_\theta}, \quad \boldsymbol{z}:[t_0,t_f]\to\mathbb{R}^{n_w\cdot n_\theta},\\
& \boldsymbol{u}:[t_0,t_f]\to\mathbb{R}^{n_u}, \quad \boldsymbol{p}\in\mathbb{R}^{n_p},\\
\text{s.t.} \quad & \boldsymbol{F}(\dot{\boldsymbol{x}}(t),\boldsymbol{x}(t),\boldsymbol{w}(t),\boldsymbol{u}(t),\boldsymbol{p},\boldsymbol{\theta})=\mathbf{0}, && \text{(11b)}\\
& \boldsymbol{F}_0(\dot{\boldsymbol{x}}(t_0),\boldsymbol{x}(t_0),\boldsymbol{w}(t_0),\boldsymbol{u}(t_0),\boldsymbol{p},\boldsymbol{\theta})=\mathbf{0}, && \text{(11c)}\\
& (10), && \text{(11d)}\\
& \boldsymbol{g}(\dot{\boldsymbol{x}}(t),\boldsymbol{x}(t),\boldsymbol{w}(t),\boldsymbol{u}(t),\boldsymbol{p},\boldsymbol{\theta})\le\mathbf{0}, && \text{(11e)}\\
& \forall t\in[t_0,t_f],
\end{aligned}$$

which is a special case of (4).

This framework has been implemented in JModelica.org. Modelica is used to encode (11b) and (11c). Optimica is used to encode (11e). The Modelica and Optimica code is then transferred to CasADi Interface by the JModelica.org compiler. A new parametric sensitivity framework has been implemented as a part of the Python side of CasADi Interface which takes the symbolic representation of (11b) and (11c) and a list of the names of the parameters in $\boldsymbol{p}_s$ and augments the DOP with (11d). Users can then use CasADi Interface and the additional sensitivity variables to construct a general DOP involving parametric sensitivities. This however requires some familiarity with CasADi. Therefore, to streamline the solution of OED problems in JModelica.org, the parametric sensitivity framework also supports the formulation of (11) by simply providing the list of names of measured variables y_i, parameters to estimate p_s, the experiment variance matrix σ, the measurement time points t_i, and the OED optimality criterion, such as A-optimality.

IV. Examples

We demonstrate the presented framework on two OED examples. The first is a simple fed-batch reactor well studied in literature, for which we compare the performance of the framework with three other frameworks for OED. The second example is for a plate-fin heat exchanger.

A. Fed-batch reactor

In the first example, we illustrate the application of OED to a biomass fermentation process, as presented in several papers as a benchmark example for the model-based metholdogy [1]–[3], [23]. The growth rate of the biomass is modeled using Monod-type kinetics, where θ_1 is the maximum specific growth rate, and θ_2 is the Monod constant. The cell death rate is linear with respect to the biomass concentration, where θ_4 is an empirical constant for the cell death. The yield coefficient of biomass to substrate is represented as θ_3 and assumed unknown. This process is carried out in a fed-batch reactor with a time-varying feed rate, where u_1 is the flow velocity (h^{-1}) and u_2 is the substrate concentration (g/L). The design equations for the biomass and substrate concentrations (y_1,y_2)

are shown below:

$$\dot{y}_1 = (r_m - u_1 - \theta_4)y_1, \qquad y_1(0) = 7, \tag{12a}$$

$$\dot{y}_2 = \frac{r_m y_1}{\theta_3} + u_1(u_2 - y_2), \quad y_2(0) = 0, \tag{12b}$$

$$r_m = \frac{\theta_1 y_2}{\theta_2 + y_2}. \tag{12c}$$

The objective is to find the experimental conditions that will allow for precise estimation of the four parameters $\theta_i, i = 1, \ldots, 4$. The set of conditions that characterize this particular experiment are the dilution factor u_1 [range 0.02–0.5 h^{-1}], and the substrate concentration in the feed u_2 [range 5–35 g/L]. The duration of the experiment is set to 20 hr. It is assumed that there are 5 sampling points, equally spaced over the experiment duration. The inputs are modeled as piecewise constant functions over five control action intervals. For simplicity, the number of sampling times and control actions are the same. The experimental variance matrix is $\sigma = 5I$. A nominal set of values of $\theta_i = 0.1$ is used to start the experiment design algorithm.

When employing simultaneous discretization, an initial guess is needed for all system variable trajectories, not only those generating degrees of freedom (inputs). With the augmented problem formulation (11), this includes the parametric sensitivities of the state and algebraic variables. To generate such initial guesses, the system is simulated in JModelica.org with CVODES (as described in Section II-D) with the constant inputs $u_1 = 0.1$ and $u_2 = 15g/L$. Using this initial guess, and a collocation discretization of 20 elements with 4 collocation points per element, the solution in Figure 2 is obtained. The NLP has 1.7×10^3 variables and is solved in 0.9 seconds with a tolerance of 10^{-8} and using the linear solver MA27 [24] in IPOPT.

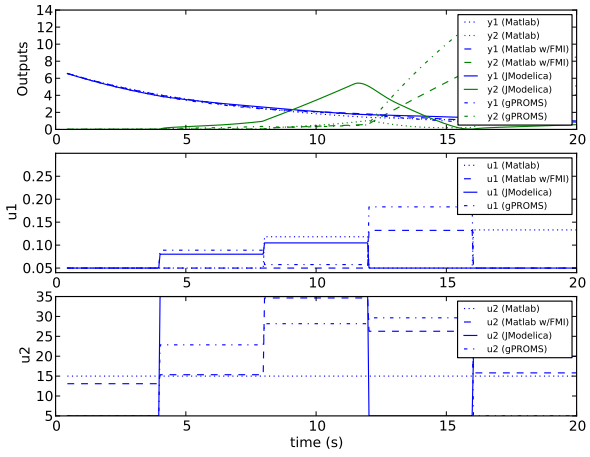

Figure 2. A-optimal experimental design for the fed-batch reactor with measurement and control sampling frequency 0.25 Hz. The problem has been solved using 4 different frameworks.

Other software optimization tools were tested with this example to compare the computational efficiency of the simultaneous discretization. The example was also solved with gPROMS and a MATLAB-based approach, either using FMI or encoding the model directly in MATLAB. gPROMS is state-of-the-art software for conducting optimal experimental design and uses single or multiple shooting. To solve for the optimal design in MATLAB, a series of scripts were configured using the Sundials solver CVODES to compute the parametric sensitivities using finite differences. The first setup contained the fed-batch reactor model in the MATLAB code, while the second setup acquired the model from an FMU file made with Modelica, imported through FMI-Toolbox [25]. This was done to observe the effects on performance and solving time when MATLAB must continuously draw from an external source. For both cases, the optimization problem was solved using a sequential quadratic programming method from MATLAB's optimization toolbox.

Table I. COMPARISON OF SOLVER TIME AND OPTIMAL DESIGN OUTPUTS FOR TESTED SOFTWARE TOOL CHAINS

Framework	Solver Time [s]	A-Optimal Value
JModelica.org	0.88	0.091
MATLAB	28.6	0.169
MATLAB w/FMI Toolbox	6040	0.168
gPROMS	3.05	0.151

Each framework uses a different set of solvers to achieve optimal design. After several iterations of this case study at various initial values, it was observed that each tool chain ends the operation at different local minima, such as in Figure 2. The optimal design with the smallest A-optimal value (from the JModelica framework) was placed into the other frameworks and it was confirmed that the A-optimal value was consistent for all four tools. It is currently not known why the output from the JModelica framework was more effectively minimized than the other frameworks. Expected variance is known to affect the optimization process, so future analysis for this framework comparison needs to take this into account, along with other possible factors.

All results for this analysis were generated on a HP Z820 Workstation (Intel® Xeon® CPU E5-2367 v2, 3.50 GHz). Table I shows that the solver times for gPROMS and JModelica.org are significantly smaller than the MATLAB and MATLAB/FMU hybrid tests. The manually written code used in the MATLAB coding significantly was less efficient than the setup for the faster programs. This is especially clear in the FMI Toolbox setup, when the function had to call from an external C code at every iteration. gPROMS already had an experimental design setup internalized before this test, so design optimization was fairly quick.

B. Heat exchanger

Another OED problem is formulated for a counter-current plate-fin heat exchanger using the same A-optimality criterion. This example comes from a marine-gas intercooler, with geometry obtained from [26]. The heat exchanger model used is more complex than the fed-batch reactor shown previously, for it contains multiple state-dependent correlations and relies on numerous structural parameters. The model consists of three states and three algebraic variables. It is assumed that the inlet gas temperature, $T_{a,\text{in}}$, is the controllable input with lower and upper bounds at 100 °C and 200 °C respectively, and the system gas and coolant (water) temperatures T_a and T_b are the states. In systems where fluid is treated upstream, it is likely that no sensors are available immediately before entering the

heat exchanger. Due to this uncertainty of inlet conditions, the entering coolant temperature $T_{b,\text{in}}$ and mass flow W_b are the parameters that need to be estimated, with nominal values 20 °C and 20 kg/s respectively. These values were originally at 20 °C and 200 kg/s for the intercooler example from Zhao et al [26], but a slower flow rate was used for this example in order to observe dynamic response). The experiment duration t_f is 200 seconds, with measurements every 5 seconds. The experimental variance matrix was $\theta = 2I$, and for simplicity's sake the system is assumed to start at steady-state.

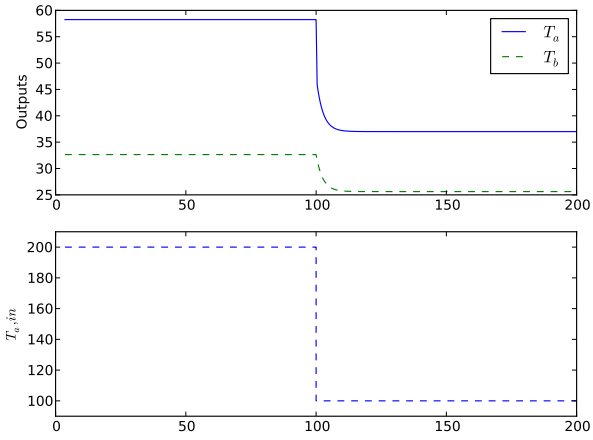

Figure 3. A-optimal experimental design for the plate fin heat exchanger with measurements every 5 seconds and a control sampling period of 100 seconds.

The problem was solved with 60 elements and 4 collocation points per element and the initial guess $T_{a,\text{in}} = 150$. The problem was solved in 5.25 seconds, with the solution shown in Figure 3. The optimal design is straightforward, as it demonstrates how the outlets behave under each specified boundary as well as through transient response, examining steady-state and dynamic heat transfer. The exiting temperatures are strongly dependent on the flow conditions hot and cold intlet streams. If the inlet temperature of one stream is different than from what is expected at steady-state then they may be undetectable, as similar outputs can be obtained through a change in either flow rate. However, by adjusting the controllable inlet temperature to achieve two separate steady-state conditions, one at each bound assigned in the design space, we can observe the effects of that change in the measurements, and more effectively characterize the expected behavior of the heat exchanger. Estimating dynamic response can also increase overall certainty for heat exchanger inlet conditions, so long as a feasible model is used. It was observed that at high flow rates, steady-state comparisons were more practical for estimation, as transient effects were too quick for observation in the sampling frequency that was used.

V. Discussion

It is noteworthy that the use of IPOPT to solve (4) assumes twice continuous differentiability of all the expressions, including $\boldsymbol{F}$, with respect to all of the optimization variables. However, since (11d) already involves first-order derivatives of $\boldsymbol{F}$, the solution of (11) will require third-order derivatives of $\boldsymbol{F}$.

The dynamic optimization framework in JModelica.org using simultaneous discretization is designed for systems of moderate complexity and size. A very rough quantification of the limitations is that the number of equations in the DAE system (4b) does not exceed 10^3 in order of magnitude. If it does, one can expect computational times and memory consumption that exceeds the capabilities of standard, modern computers. This means that for the solution of (11), $(n_x + n_w) \cdot n_\theta$ should not exceed 10^3 in order of magnitude. For problems of this size, simultaneous methods often outperform shooting-based methods when combined with algorithmic differentiation rather than finite differences, both in terms of speed and reliability.

The OED problem (11) is a rather simple variant of OED. Possible generalizations would be to consider multiple experiments, free measurement time points t_i, and free time horizon endpoints t_0 and t_f. The implementation of multiple experiments is straightforward by simply duplicating the DAE and forward sensitivity equations for each experiment, meaning that for n_e experiments, the number of equations in the transformed DAE system will be $(n_x + n_w) \cdot (1 + n_\theta) \cdot n_e$. For systems with many algebraic variables, this number can be greatly reduced by eliminating the algebraic variables using the techniques presented in [27].

Support for free measurement time points would require a more significant implementation effort, with two distinct possible approaches. The first approach is to implement support for multiphase problems [5, Section 3.7] and having one phase per measurement point. Due to the difficulty of solving multiphase problems with free phase endpoints, a seemingly better approach is to use global collocation, allowing for the evaluation of the sensitivities at free time points using the global collocation polynomials. Once support for free measurement time points is available, the extension to free experiment durations comes for free, as the JModelica.org framework already supports dynamic optimization with free time horizon endpoints.

The paper has focused on A-optimal design. Extending the framework to handle D-optimal design would be straightforward. However, the framework in JModelica.org relies heavily on differentiability, which makes it less straightforward to support non-differentiable criteria such as E-optimality. The E-criterion can however be reformulated to get differentiability by using semidefinite programming [28].

The solutions obtained for the fed-batch reactor example in Section IV are not global optima. That example has many local optima, and global optimization techniques should be utilized if the global optimum is sought. The presented framework lends itself well to global optimization techniques which are based on locally solving subproblems, such as the simple approach of Latin hypercube sampling [29] of the degrees of freedom.

VI. Conclusion

Dynamic parametric sensitivity optimization problems, in particular model-based optimal design of experiments, are typically solved numerically using shooting methods. In this paper

we proposed a method for solving these problems using simultaneous discretization, based on forward sensitivity analysis, which is feasible and efficient for systems of moderate size. We also presented the implementation of a high-level framework in JModelica.org to address a relevant gap in the capabilities for dynamic optimization and parameter estimation offered by Modelica tools. The implementation was demonstrated on two examples and its performance was compared to equivalent gPROMS- and a MATLAB- based solution approaches. It was shown that the proposed framework is computationally efficient and precise. It was also demonstrated that the proposed tool chain allows for seamless integration of the programmatic environments that perform dynamic optimization of parametric sensitivities. Possible extensions of the proposed method and framework were also discussed, in particular the treatment of free measurement time points and experiment durations.

ACKNOWLEDGMENT

Fredrik Magnusson acknowledges support from the Swedish Research Council through the LCCC Linnaeus Center and is a member of the eLLIIT Excellence Center at Lund University. The remaining authors were sponsored by the UTC Institute for Advanced Systems Engineering (UTC-IASE) of the University of Connecticut and the United Technologies Corporation. Lu Han also acknowledges support by the National Science Foundation under Grant No. 1054718. Any opinions expressed herein are those of the authors and do not represent those of the sponsors. Help and guidance by Modelon and Modelon AB are gratefully acknowledged.

REFERENCES

[1] F. Galvanin, S. Macchietto, and F. Bezzo, "Model-based design of parallel experiments," *Ind. Eng. Chem. Res.*, vol. 46, pp. 871–882, 2007.

[2] S. Asprey and S. Macchietto, "Statistical tools for optimal dynamic model building," *Comput. Chem. Eng.*, vol. 24, pp. 1261–1267, 2000.

[3] ——, "Designing robust optimal dynamic experiments," *J. Process Control*, vol. 12, pp. 545–556, 2002.

[4] L. T. Biegler, *Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes*. Philadelphia, PA, USA: Mathematical Optimization Society and the Society for Industrial and Applied Mathematics, 2010.

[5] J. T. Betts, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*, 2nd ed. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2010.

[6] E. Balsa-Canto, A. A. Alonso, and J. R. Banga, "Computational procedures for optimal experimental design in biological systems," *IET Systems Biology*, vol. 2, pp. 163–172, 2008.

[7] I. Bauer, H. G. Bock, S. Körkel, and J. P. Schlöder, "Numerical methods for optimum experimental design in DAE systems," *J. Comput. Appl. Math.*, vol. 120, pp. 1–25, 2000.

[8] G. Franceschini and S. Macchietto, "Model-based design of experiments for parameter precision: State of the art," *Chem. Eng. Sci.*, vol. 63, pp. 4846–4872, 2008.

[9] P. Fritzson, *Principles of Object-Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical Approach*, 2nd ed. Piscataway, NJ, USA: Wiley-IEEE Press, 2015.

[10] J. Åkesson, K.-E. Årzén, M. Gäfvert, T. Bergdahl, and H. Tummescheit, "Modeling and optimization with Optimica and JModelica.org—languages and tools for solving large-scale dynamic optimization problems," *Comput. Chem. Eng.*, vol. 34, pp. 1737–1749, 2010.

[11] M. Hoang, T. Barz, V. Merchan, L. Biegler, and H. Arellano-Garcia, "Simultaneous solution approach to model-based experimental design," *AIChE J.*, vol. 59, pp. 4169–4183, 2013.

[12] S. Mayr, G. Grabmair, and J. Reger, "Input design and parameter estimation with open source tools," in *8th Vienna International Conference on Mathematical Modelling*, Vienna, Austria, 2015.

[13] S. E. Mattsson and G. Söderlind, "Index reduction in differential-algebraic equations using dummy derivatives," *SIAM J. Sci. Comput.*, vol. 14, pp. 677–692, 1993.

[14] D. Liberzon, *Calculus of Variations and Optimal Control Theory: A Concise Introduction*. Princeton, NJ, USA: Princeton University Press, 2012.

[15] F. Magnusson and J. Åkesson, "Dynamic Optimization in JModelica.org," *Processes*, vol. 3, pp. 471–496, 2015.

[16] R. Serban and A. C. Hindmarsh, "CVODES: The sensitivity-enabled ODE solver in SUNDIALS," in *In Proceedings of the ASME 2005 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 2005.

[17] J. Åkesson, "Optimica—an extension of Modelica supporting dynamic optimization," in *In Proceedings of the 6th International Modelica Conference, Bielefeld, Germany, 2008*.

[18] B. Lennernäs, "A CasADi based toolchain for JModelica.org," M.Sc. thesis, Lund University, Sweden, 2013.

[19] A. Wächter and L. T. Biegler, "On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, pp. 25–57, 2006.

[20] J. Andersson, "A general-purpose software framework for dynamic optimization," Ph.D. thesis, Arenberg Doctoral School, KU Leuven, 2013.

[21] T. Blochwitz, M. Otter, M. Arnold, C. Bausch, C. Clauß, H. Elmqvist, A. Junghanns, J. Mauss, M. Monteiro, D. Neumerkel, and J.-V. Peetz, "The functional mockup interface for tool independent exchange of simulation models," in *Proceedings of the 8th International Modelica Conference*, Dresden, Germany, 2011.

[22] C. Andersson, C. Führer, and J. Åkesson, "Assimulo: A unified framework for ODE solvers," *Math. Comput. Simulat.*, 2015, in press.

[23] D. Espie and S. Macchietto, "The optimal design of dynamic experiments," *AIChE J.*, vol. 35, pp. 223–229, 1989.

[24] HSL, "A collection of Fortran codes for large scale scientific computation," Software available at http://www.hsl.rl.ac.uk.

[25] Modelon, "FMI Toolbox for MATLAB/Simulink," Software available at http://www.modelon.com/products/fmi-toolbox-for-matlab/.

[26] N.-b. Zhao, X.-y. Wen, and S.-y. Li, "Dynamic time-delay characteristics and structural optimization design of marine gas turbine intercooler," *Math. Prob. Eng.*, vol. 2014, 2014.

[27] F. Magnusson, K. Berntorp, B. Olofsson, and J. Åkesson, "Symbolic transformations of dynamic optimization problems," in *Proceedings of the 10th International Modelica Conference*, Lund, Sweden, 2014.

[28] L. Vandenberghe and S. Boyd, "Semidefinite programming," *SIAM Review*, vol. 38, pp. 49–95, 1996.

[29] M. D. McKay, R. J. Beckman, and W. J. Conover, "Comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21, pp. 239–245, 1979.