IDOS - (also) a Web Based Tool for Calibrating Modelica Models[*]

Radoslaw Pytlak Tomasz Tarnawski
Warsaw Technical University, Institute of Automatic Control and Robotics
ul. Sw Andrzeja Boboli 8, 02-525 Warsaw

Abstract

This paper presents a newly deployed server, IDOS, an online-accessible environment providing the service of solving optimal control problems. Development and deployment of the Interactive Dynamic Optimization Server is a result of projects funded by NCBiR (National Center for Research and Development). One of the outcomes of the project was a modeling language (Dynamic Optimization Modeling Language, DOML) providing a uniform format for defining dynamic optimization problems. DOML is an extension of Modelica language and hence, not only a user can specify his problem in the way he does in Modelica but also (more importantly, for the purpose of this paper) models created in Modelica for simulation purposes can be easily transferred to DOML for solving their related optimization problems. In particular, Modelica models can be calibrated with the help of our server. The paper tries to illustrate the point in depth. It presents the workings of the server and reviews the scope of solvers implemented, focusing especially on those that can be used for calibrating Modelica models. Special attention is devoted to an algorithm using adjoint equations for evaluating sensitivities of model equations with respect to parameters and to calibrating models described by higher index DAEs.

Keywords: dynamic optimization; optimal control; model calibration

1 Introduction

The paper describes computing environment IDOS accessed by Internet and created for solving optimal control problems. The IDOS server is equipped with its Dynamic Optimization Modeling Language (DOML) for defining optimal control problems. The DOML format provides a mean for describing an optimal control problem in slightly extended Modelica syntax. It is very similar to Optimica ([1],[2]) and in fact its implementation is based on the JModelica.org Optimica compiler, although a number of features proposed in DOML set it apart from Optimica (these are described in detail in the accompanying paper [24]).

The IDOS server was created to provide means for solving optimal control problems by essentially different solvers such as: multiple shooting methods; methods based on *a'priori* discretization of systems equations and then transforming the problem to a large–scale nonlinear programming problems; methods which use adjoint equations in order to evaluated reduced gradients of functionals defining the problem. From the beginning we have followed the idea that the server should enable a strategy of chaining of solvers–starting with some solver to get a rough approximation to the optimal control solution and then proceeding further with another solver which completes the solution process by giving an accurate solution to the problem. Having this in mind we knew from the beginning that the server should have as many different solvers as possible. Since all these solvers were to be invoked from the DOML script this approach enabled us to provide quite precise requirements for the dynamic optimization language. In the paper [24] we describe how that approach has been materialized in the form of new constructs put into the Modelica framework.

In the following sections the paper briefly describes the IDOS server by paying much attention to the types of optimal control problems which are supported by the server. Then we describe in some details numerical procedures which are avail-

[*]The work presented in the paper has been partially supported by NCBiR grants: R02-0009-06, PBS1/A7/6/2012

able at the IDOS server and which can be used to calibrate various Modelica type problems. These solvers constitute a subset of all solvers available at IDOS which are also mentioned in the paper.

2 The IDOS server

The main task of the IDOS platform ([20],[19],[21],[26]) is to manage the processes of solving the dynamic optimization problems. Number of tasks that can be solved over a period of time depends on the server CPU. When its computing power is insufficient, optimization task are queued to be executed when the server resources are released. The solving process on IDOS is build of the general steps: 1) define the optimization problem using dedicated language (DOML, Dynamic Optimization Modeling Language); 2) submit the problem to IDOS platform; 3) interpret obtained results.

During the design process of the IDOS platform the following assumptions were made: 1) the primary goal of the system is to allow users to define, calculate and view results of optimization tasks; 2) all functions of the system will be available remotely via the Internet, without the need to install dedicated software on the user's machine. Different tools used to developed web applications will be used to fulfill this task; 3) the system is designed to manage several types of optimization libraries; 4) the system design is modular so it can be expanded and launched in stages (for example we can extend the system by adding computation servers or implementing support for newly developed numeric libraries); 5) the system should include user management module (registration, login). Each user should be able to define, calculate their own optimization tasks (within own account on the IDOS platform); 6) the results of optimization calculations shall be presented in a universal, unified form.

Each submitted optimization problem is processed in a pipeline fashion, beginning with a task data collection from a database and ending with storing the results. After a task is stored in a database and its status set as ready problem is located by a Data Base Adapter daemon (a process contunously running on the server). Next prepared task is analyzed by the DOML Compiler and in result C++ source code is generated (or compilation errors are reported if the submitted definition was faulty). In the following step a C++ code is compiled. The final binary file is built on the base of optimization libraries stored on the IDOS server. The last step is the execution of the build binaries. When the job is finished, results are stored in the Central Archive database. If some error appeared its description is stored in the Central Configuration Database and the status of a task is set as failed. The optimization results are stored in uniquely named XML files. The name and location of each such file is stored in a central database.

3 Solvers implemented and libraries used

The IDOS server has been built to solve primarily the following optimal control problem

$$\min_u \phi(x(t_f)) \tag{1}$$

subject to the constraints:

$$\begin{aligned} F(\dot{x},x,u,t) &= 0 \text{ a.e. } t \in T = [0,t_f] && (2)\\ q(x(t),t) &\leq 0,\ t \in T && (3)\\ h_i^1(x(t_k)) &= 0,\ i \in E && (4)\\ h_j^2(x(t_k)) &\leq 0,\ j \in I && (5)\\ u(t) &\in \Omega \text{ a.e. } t \in T. && (6)\end{aligned}$$

In general we assume that the structural index of equations (2) does not exceed three ([16]). If differential–algebraic equations in the above problem reduce to ODEs then more general problems (with respect to constraints) can be handled by the server.

At the moment the IDOS server can handle control problems described with ordinary equations and differential–algebraic equations but the incorporation into the IDOS solvers for problems with partial differential equations is also under way.

The IDOS server enables solving optimal control problems by using essentially different methods:

a) based on an *a'priori* discretization of systems equations–optimal control problem is then transformed to a large scale nonlinear programming problem;

b) based on numerical methods for integrating system equations with variable stepsizes–reduced gradients are then evaluated with the help of the corresponding adjoint equations;

c) based on shooting methods applied to differential equations derived from necessary optimality conditions for optimal control problems;

d) based on adjoint equations for problems described by higher index differential–algebraic equations;

e) based on Mixed Integer Programming algorithms for optimal control problems with ordinary differential equations in which some control variables take only integer values.

Obviously, not all solvers can be used on the every problem. Their applicability is briefly discussed below.

3.1 Optimal control problems described by ODEs

As far as optimal control problems described by ODEs are concerned essentially three different approaches to solving them are applied.

In the first approach system equations are discretized *a'priori* (with respect to time) and in effect replaced by nonlinear algebraic equations. As a result, a large scale nonlinear programming problem (NLP) is solved. The main library used in this case is the `OLADO` package ([3]) which contains a wide range of optimization solvers, mostly based on some version of the SQP algorithm. Furthermore, in the `OLADO` package three different interior point QP solvers are available: procedure which implements Mehrotra's primal-dual predictor-corrector method [13]; procedure based on the Mehrotra's algorithm modified by R. Franke [6]; and the procedure which applies the Gondzio's multiple centrality correctors variant of the primal-dual interior-point method for convex QP [8].

The `IDOS` server is also equipped with sparse nonlinear optimization `HQP` solver and its `OMUSES` interfaces to ODEs integrators ([7]). In particular, the `HQP` solver is linked with: procedures employing Euler, or simple Runge–Kutta fourth order rules (RK4); `GRK4` procedure of Rosenbrock's type ([11]); `DOPRI5` procedure based on explicit Runge–Kutta scheme due to Dormand and Prince ([10]); `IMP` procedure which uses the implicit midpoint rule ([11]); `SDIRK` procedure based on singly diagonally implicit Runge–Kutta formula ([11]) and with `ODETS` procedure which uses Taylor's expansion of ODEs derived from `ADOL-C` driver routine `forodec`.

The second approach 'preserves' the continuous time of the optimal control problem to be solved. It means that during numerical treatment the system's equations are integrated by a procedure with variable stepsizes. Since state variables are not decision variables the problem is considered in the, so called, reduced space, i.e. all functionals defining the problem are functionals of control variables only —gradients of these functionals, required by an optimization procedure, are evaluated with the help of the adjoint equations which are consistent with the system equations.

Essentially, there are two procedures on the `IDOS` server which follow this approach. In the first one, system equations are integrated by procedures from the `cvodes` part of the `SUNDIALS` package ([22]), also adjoint equations are integrated by procedures from the `cvodes`. Iteration of the optimization procedure is performed by `IPOPT` ([25]). In the second procedure, implicit Runge–Kutta method is employed for system equations integration and adjoint equations are evaluated in accordance with the discretization of ODEs by the integration procedure. In this case optimization is performed by the SQP method which applies an active set strategy ([17]). The procedure described in [17] was designed to cope efficiently with control problems with state constraints.

The third approach to optimal control problems with ODEs deals with multiple shooting methods. At the moment within the `IDOS` server an indirect shooting method based on Oberle's code ([14]) is implemented.

3.2 Optimal control problems described by DAEs

The server currently offers two procedures for solving optimal control problems with DAEs. The first one is aimed at problems whose differential–algebraic equations have index one. It is based on the implicit Runge–Kutta integration procedure but adopted to index one DAEs in the semi–explicit form ([17]). The optimization engine used here is the SQP procedure which uses an active set strategy in corresponding QP subproblems (a range–space variant of it).

The second procedure carries out the unique approach to optimal control problems with higher index DAEs described in [18]. The approach does not require the system to be transformed from DAEs to ODEs (through differentiation, with respect to time, of selected algebraic equations). Instead, the constructed procedure employs an implicit Runge–Kutta method for system equations integration. We use Radau IIA method as implemented in [11] (see also [9]). Reduced gradients are evaluated on the basis of adjoint equations defined for discretized (by the integration procedure) system equations. The procedure uses the algorithm for consistent initialization of system equations and for that purpose the method based the Pantelides' algorithm ([15]) and the kinsol procedure from the SUNDIALS package was implemented. To our knowledge, it is the first successfully implemented procedure for solving optimal control problems described by higher index DAEs which does not need transformation of the DAEs to the underlying ODEs.

3.3 Optimal control problems with integer valued controls

For the moment, optimal control problems with integer valued controls solved at the IDOS server can only have ordinary differential equations. For these problems we used BONMIN package (see e.g. [4], [5]). It has been integrated with the OLADO library and with the procedures based on the cvodes procedures from SUNDIALS package.

Table 1 summarizes to some extent the solvers structure of the IDOS server — in fact it is much more complex since, for instance, procedures such as cvodes, or BONMIN contain themselves a number of essentially different methods which can be accessed by setting their parameters. Furthermore, the server employs ADOL-C package for performing automatic differentiation and OpenBLAS package for linear algebra operations.

The code developed for the server is meant as open source[1] (most of the libraries used by the server are open-source, as well, see Table 1), hence one could deploy own instances of IDOS server, at least in principle. The experience teaches, however, that installation, configuration and maintenance of all required packages is in itself far from trivial. For that reason, at this point there is no dedicated distribution package for automated installation of the server suite on user machines, although development of such is planned.

Method and problem type	Solvers
a'priori discretization, ODEs	OLADO: SQP (Powell's type) + Euler IPOPT + Euler HQP + Euler HQP + OMUSES: HQP + Euler HQP + DOPRI5 HQP + GRK4 HQP + IMP HQP + ODETS HQP + SDIRK HQP + RK4
adjoint equations, ODEs	RKCON + Radau IIA IPOPT + SUNDIALS (cvodes)
shooting method, ODE	BNDSCO + RK4
a'priori discretization, ODEs (integer controls)	BONMIN + Euler SQP (Powell's type) + Euler
adjoint equations, ODE (integer controls)	BONMIN + SUNDIALS (cvodes)
adjoint equations, DAEs (index = 1)	RKCON + Radau IIA IPOPT + SUNDIALS (idas) - u. dev.
adjoint equations, DAEs (index ≤ 3)	RKCON + RADAU5 + SUNDIALS (kinsol) + MAXIMA

Table 1: Listing of IDOS main solvers

4 Estimation of models parameters

Suppose that we have the dynamic model with parameters p:

$$\dot{x}(t) = f(x(t), t, p),\ t \in [0, t_f],\ x(0) = x_0. \quad (7)$$

which solution x^p is dependent on parameters p. It is often the case that such model describes a real life, observable phenomenon but the exact values of its parameters are unknown. Then, they could be approximated by solving nonlinear least squares problem:

$$\min_p \sum_{k=1}^{N} \left[(x_l(t_k) - x_l^e(t_k))^2 \right] \quad (8)$$

subject to the constraints (7), where x_l^e is an empirical (measured) trajectory.

Since x is a function of parameters p, the problem can be stated as

$$\min_p \left[F(p) = \sum_{k=1}^{N} \left(x_l^p(t_k) - x_l^e(t_k) \right)^2 \right] \quad (9)$$

[1]contact the authors for setting up access to code repository

That problem can be solved by nonlinear programming techniques provided that we can evaluate gradients of $F(p)$. In particular, we can apply the Gauss–Newton algorithm (see section 5) to problem (9) taking advantage of the least–squares structure of ith objective function.

The gradient of the functional $F(p)$ may be evaluated with the help of adjoint equations if we observe that $F(P)$ is composed of N functions $F^k(p)$:

$$F^k(p) = (x_l^p(t_k) - x_l^e(t_k))^2$$

where x^p is the solution to the equations (7).

The gradient of $F^k(p)$ is given by the formula:

$$\nabla F^k(p) = \int_0^{t_k} f_p(x(t),t,p)^T q_k(t)dt \qquad (10)$$

where q_k is the solution to the adjoint equations

$$\begin{array}{rcl} q_k(t_k) & = & Q_k \\ \dot{q}_k(t) & = & -f_x(x(t),t,p)^T q_k(t), \ t \in [0,t_k). \end{array}$$

Here,

$$Q_k = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 2(x_l(t_k) - x_l^e(t_k)) \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Then $\nabla F(p) = \sum_{k=1}^N \nabla F^k(p)$.

Having objective function values and its gradients we can build an optimization procedure for model calibration. A general scheme for these procedure may look like stated below:

General Calibration Procedure

1. Set initial values of parameters: p_1 and set $k=1$.
2. For parameters p_k calculate system trajectories x^{p_k} by numerically integrating system equations using, for example, procedure cvodes from `SUNDIALS` package. On that basis determine objective function value $F(p_k)$ through values $F^l(p_k)$, $l = 1,\ldots,N$.
3. Having trajectories x^{p_k} and values $F^l(p_k)$, $l = 1,\ldots,N$ solve adjoint equations (using, for example procedure cvodes), and determine $\nabla F^l(p_k)$, $l = 1,\ldots,N$s and $\nabla F(p_k)$.
4. Determine the direction of descent using some optimization procedure (for example, `Ipopt`).
5. Perform directional minimization with the help of optimization procedure to evaluate step size α_k. Substitute $p_{k+1} = p_k + \alpha_k d_k$. increase k by one and go to Step 2).

5 Gauss–Newton method for the least squares problem

One approach to implementating the just-outlined general calibration procedure makes use of the Gauss–Newton method for solving nonlinear least squares problems. For the purpose of discussing it, consider the following optimization problem:

$$\min_{x \in \mathcal{R}^n} \left[f(x) = \frac{1}{2}\|g(x)\|^2 = \frac{1}{2}\sum_{i=1}^m \|g_i(x)\|^2 \right] \qquad (11)$$

where

$$g(x) = \begin{bmatrix} g_1(x) \\ g_2(x) \\ \vdots \\ g_m(x) \end{bmatrix}. \qquad (12)$$

In this case, the gradient and the Hessian matrix of the objective function (11) are defined as follows:

$$\begin{array}{rcl} \nabla f(x) & = & \sum_{i=1}^m g_i(x)\nabla g_i(x) = J(x)^T g(x) \\ \nabla^2 f(x) & = & \sum_{i=1}^m \nabla g_i(x)\nabla g_i(x) + \\ & & \sum_{i=1}^m g_i(x)\nabla^2 g_i(x) \\ & = & J(x)^T J(x) + \sum_{i=1}^m g_i(x)\nabla^2 g_i(x). \end{array}$$

Here, $J(x)$ is the Jacobian matrix of the transformation matrix (12). Assuming that $g_i(x)\nabla^2 g_i(x) \approx 0$ we can write

$$\nabla^2 f(x) \approx J(x)^T J(x).$$

The Gauss–Newton method minimizes, at every iteration, linearization of g at the point x_k:

$$\tilde{g}(x_k;x) = g(x_k) + J(x_k)(x - x_k).$$

The next point x_{k+1} is obtained by solving the problem

$$\min_{x\in\mathcal{R}^n} \frac{1}{2}\|\tilde{g}(x_k;x)\|^2.$$

Then

$$x_{k+1} = x_k - \left(J(x_k)^T J(x_k)\right)^{-1} J(x_k)^T g(x_k) \quad (13)$$

It is a descent method since

$$-J(x_k)^T g(x_k)$$

is a descent direction for the function $\frac{1}{2}\|g(x_k)\|^2$, because

$$\nabla\left(\frac{1}{2}\|g(x_k)\|^2\right) = J(x_k)^T g(x_k)$$

and provided that

$$\left(J(x_k)^T J(x_k)\right) > 0$$

(it means matrix positiveness).

The outlined algorithm is most often applied with the modifications

$$\begin{aligned} x_{k+1} = x_k - \alpha_k \left(J(x_k)^T J(x_k) + \Delta_k\right)^T \times \\ \times J(x_k)^T g(x_k), \qquad (14) \end{aligned}$$

where $\alpha_k > 0$ is the result of the directional minimization, and Δ_k is such diagonal matrix that

$$\left(J(x_k)^T J(x_k) + \Delta_k\right) > 0.$$

If the least squares problem is used to calibrate a model then on the basis of experimental data $\{y_i, z_i\}_1^m$ we evaluate models parameters x

$$z = h(y,x).$$

In this case models parameters are calculated by solving the optimization problem

$$\min_x \left[f(x) = \frac{1}{2}\sum_{i=1}^{m} \|z_i - h(y_i,x)\|^2\right].$$

The Gauss–Newton approach was implemented along the lines of the *General Calibration Procedure* presented in Section 4. In Step 4) of the procedure, the direction of descent d_k is determined on the basis of vectors $\nabla F^l(p_k)$, $l = 1,\ldots,N$ and $\nabla F(p_k)$. By referring to relations (13)–(14) we set

$$\begin{aligned} J(p_k) &= \begin{bmatrix} \nabla F^1(p_k) \\ \nabla F^2(p_k) \\ \vdots \\ \nabla F^N(p_k) \end{bmatrix} \\ H(p_k) &= J(p_k)^T J(p_k) \\ p_{k+1} &= p_k - \alpha_k [H(p_k)]^{-1} \nabla F(p_k). \quad (15) \end{aligned}$$

The formula (15) uses the stepsize α_k which usually is evaluated with the help of sophisticated procedures. We decided not to build new optimization packages but to adopt existing ones therefore α_k is determined according to `Ipopt` or `RKCON` procedures. Similarly, scaling matrices $H(p_k)$ are made nonsingular by procedures built-in `Ipopt` or `RKCON`. For example, it can be done in Ipopt by 'cheating' the method `eval-h` with matrices $H(p_k)$ instead of true Hessian matrices evaluated at p_k.

6 Estimation of parameters of higher index DAEs

The estimation procedure stated in Section 4 can also be adopted to higher index DAEs provided that numerical procedure presented in [18] is used. Suppose that our system equations are as follows:

$$F(\dot{x}(t), x(t), t, p), \; t \in [0, t_f] \quad (16)$$

and, as before, $p \in \mathcal{R}^m$ is the vector of unknown parameters.

When the integration scheme from [11] is extended to implicit equations then we will arrive at equations

$$\begin{aligned} F(x_i'(k+1), x(k) + \\ h(k)\sum_{j=1}^{s} a_{ij}x_j'(k+1), t_k + c_i h(k), p) = 0, \quad (17) \end{aligned}$$

$$x(k+1) - x(k) + h(k)\sum_{i=1}^{s} b_i x_i'(k+1) = 0, \quad (18)$$

$i = 1,\ldots,s$ and they represent the dynamics of a discrete time control system.

In order to define state equations we introduce the state vector $X(k)$:

$$X(k) = \begin{bmatrix} x_1'(k) \\ \vdots \\ x_s'(k) \\ x(k) \end{bmatrix}, \quad (19)$$

then equations (17)–(18) become

$$\tilde{F}(X(k+1),X(k),k,p)=0. \tag{20}$$

Here, $X(k)\in\mathcal{R}^{(s+1)n}$, $\tilde{F}:\mathcal{R}^{(s+1)n}\times\mathcal{R}^{(s+1)n}\times\{0,\dots,N-1\}\times\mathcal{R}^{m}\rightarrow\mathcal{R}^{(s+1)n}$.

System (20) is fully implicit discrete time and, under some nonsingularity assumption, can be expressed as explicit. If the Jacobian of $\hat{F}$ with respect to $X(k+1)$, denoted by $\tilde{F}_{X^+}$, exists and is nonsingular for all $k=0,\dots,N-1$, then from the Implicit Function Theorem there exists unique function φ such that

$$X(k+1)=\varphi(X(k),k,p) \tag{21}$$

and

$$\tilde{F}(\varphi(X(k),k,p),X(k),k,p)=0, \tag{22}$$

for $k=0,\dots,N-1$. Under differentiability assumptions imposed on F the function φ is differentiable with respect to $X(k)$ and p which means that we can write

$$\begin{gathered}\tilde{F}_{X^+}(k)\varphi_X(k)+\tilde{F}_X(k)=0\Rightarrow\\ \varphi_X(k)=-\left[\tilde{F}_{X^+}(k)\right]^{-1}\tilde{F}_X(k) \qquad (23)\\ \tilde{F}_{X^+}(k)\varphi_p(k)+\tilde{F}_p(k)=0\Rightarrow\\ \varphi_p(k)=-\left[\tilde{F}_{X^+}(k)\right]^{-1}\tilde{F}_p(k). \qquad (24)\end{gathered}$$

$\tilde{F}_{X^+}(k)$, $\tilde{F}_X(k)$, $\tilde{F}_p(k)$ are evaluated at $(X(k+1),X(k),k,p)$ and $\varphi_X(k)$, φ_p at $(X(k),k,p)$.

If we consider optimal control problem with the objective function

$$\hat{F}^0(p)=(x_l^p(t)-x_l^e(t))^2 \tag{25}$$

then the gradient of it can be calculated by referring to adjoint equations for the functional (25) and the system (21).

The adjoint equations for the functional (25) and the system (21) are considered, for example, in [17]:

$$\begin{aligned}p(t)&=Q_t &(26)\\ p(k)&=\varphi_X(k)^Tp(k+1), &(27)\end{aligned}$$

$k=0,\dots,t-1$ and

$$Q_t=\begin{bmatrix}0\\0\\\vdots\\2(x_l^p(t)-x_l^e(t))\\0\\\vdots\\0\end{bmatrix}.$$

Notice that vector Q_t has dimension $(n+1)\times s$.

Then adjoint variables p are the means for the gradient evaluation according to the formula

$$\hat{F}_p^0(p)=\sum_{k=1}^{t-1}\varphi_p(k)^Tp(k+1). \tag{28}$$

Using (23)–(24) the adjoint equations (27) and the formula (28) can be expressed without the knowledge of φ:

$$\begin{aligned}p(k)&=-\tilde{F}_X(k)^T\left[\tilde{F}_{X^+}(k)\right]^{-T}p(k+1)\\ \hat{F}_p^0(p)&=-\sum_{k=1}^{s-1}\tilde{F}_p(k)^T\left[\tilde{F}_{X^+}(k)\right]^{-T}p(k+1).\end{aligned}$$

Fortunately, the calculation of $[\tilde{F}_{X^+}(k)]^{-1}$ can be avoided with the help of additional variable r which is the solution to the linear equations

$$\tilde{F}_{X^+}(k)^Tr(k+1)=p(k+1). \tag{29}$$

Then the adjoint equations become

$$\begin{aligned}p(k)&=-\tilde{F}_X(k)^Tr(k+1) &(30)\\ \hat{F}_p^0(p)&=-\sum_{k=1}^{s-1}\tilde{F}_p(k)^Tr(k+1). &(31)\end{aligned}$$

Eventually we have the viable formula for the gradient of $\hat{F}(p)$ provided that matrices $\tilde{F}_{X^+}(k)$ are nonsingular and equations (29) can be solved efficiently.

Important observation is that matrices $\tilde{F}_{X^+}(k)$ are **nonsingular** even for index three problems provided that $h(k)$ are sufficiently small–see [18] for details. Therefore, for many higher index DAEs we have a valid technique for computing gradients of $\hat{F}^0(p)$. The evaluation of adjoint equations requires solving linear equations with matrices which have to be evaluated (and factorized) anyway while numerically integrating system equations.

The estimation procedure discussed in this section requires the program for consistent initialization at initial time. That program consists of two steps.

Consistent Initialization Procedure

1. In the first step new equations have to be determined by differentiating some equations with respect to time.
2. In the second step the extended set of equations is solved, at initial time, to get initial values of original equations.

In our approach Step 1) of *Consistent Initialization Procedure* is carried out on the basis of a graph based algorithm proposed by Pantelides ([15]). The algorithm finds the minimal subset of equations which need to be differentiated.

Next symbolic differentiation is performed. A good enough tool with symbolic differentiation module is an open source computer algebra system `Maxima` ([23]) which is descendant of `Macsyma`, the computer algebra system developed in the late 1960s at MIT. In the next section we show that we start solving optimal control problems by stating them in the `DOML` environment. In that case the first step in the symbolic differentiation is a conversion of the equations defined in the `DOML` format to the `Maxima` form. Then, the `Maxima` library is invoked and as a result differentiated equations are returned to the system console.

Eventually, in Step 2) of *Consistent Initialization Procedure*, the set of nonlinear (in general) algebraic equations is solved by using the `KINSOL` solver from the `SUNDIALS` package ([12]) (alternatively the `Ipopt` solver can be applied–[25]).

Summing up, the `IDOS` server is equipped, for the moment, with optimization procedures which either use quasi–Netwon approximations of Hessian matrices (if `Ipopt` or `RKCON` are used without modifications), or their Gauss–Newton approximations. They are based on adjoint equation evaluations and can be applied to models described by ODEs or higher index DAEs. Procedures for models calibration based on *a'priori* discretization of systems equations are under development.

7 Examples

Example 1 The first example is the chemical kinetics model available as `SUNDIALS` example. The equations of the reaction rates are as follows:

$$\begin{aligned}\dot{x}_1 &= -p_1x_1+p_2x_2x_3\\ \dot{x}_2 &= p_1x_1-p_2x_2x_3-p_3x_2^2\\ \dot{x}_3 &= p_3x_3^2\end{aligned}$$

The models transcription in `Modelica` is shown on Listing 1. The corresponding DOML file for the model calibration is presented on Listing 2. It defines the reference trajectory through the spline function of degree 1. All three parameters there are decision variables.

When $p_1 = 0.04$, $p_2 = 1.e7$ and $p_3 = 3e7$ we have the `SUNDIALS` example and we call the model with

```
package Roberts
  model Roberts_model(startTime=0.0,
                      finalTime=1.0 )
    parameter Real p1 = 0.04;
    parameter Real p2 = 1e4;
    parameter Real p3 = 3e7;
    Real x1(start = 1.0);
    Real x2(start = 0.0);
    Real x3(start = 0.0);
  equation
    der(x1) + p1*x1 - p2*x2*x3 = 0;
    der(x2) - p1*x1 + p2*x2*x3 + p3*x2*x2 = 0;
    der(x3) - p3*x2*x2 = 0;
  end Roberts_model;
end Roberts;
```

Listing 1: Roberts model in Modelica

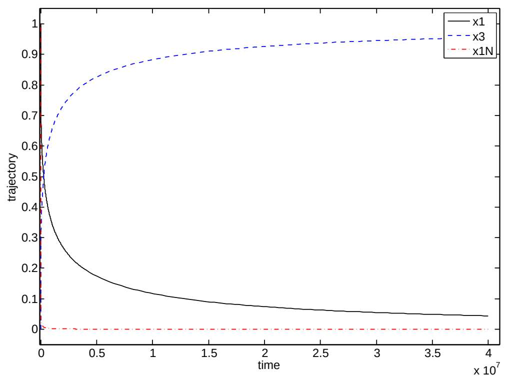

Figure 1: The calibration of Roberts model.

```
package Roberts
  import I = Modelica.DOML.Inputs;
  optimization Roberts_opt(startTime=0.0,
                           finalTime=1.0 )
    minimize AccErr = err(finalTime);
    I.Spline x1r(startTime = startTime,
      finalTime =finalTime, degree=1,
      table=[0.0, 0.6;8e6, 0.3; 16e6, 0.2;
      24e6,0.1;32e6,0.01;40e6,0.0;4.0e7,0.0]);
    parameter Real p1(free=true,
      initialGuess=0.04, min=0.0, max=10.0);
    parameter Real p2(free=true,
      initialGuess=1e4, min=0.0, max=10.0);
    parameter Real p3(free=true,
      initialGuess=3e7, min=0.0, max=10.0);
    input Real x1e(free=false) = x1r.y;
    Real x1(start = 1.0);
    Real x2(start = 0.0);
    Real x3(start = 0.0);
    Real err(start = 0.0);

    annotation(solver="nlsq_cvodes", Steps="5",
               RTOL_tolerance="1.0e-6");
  equation
    der(x1) + p1*x1 - p2*x2*x3 = 0;
    der(x2) - p1*x1 + p2*x2*x3 + p3*x2*x2 = 0;
    der(x3) - p3*x2*x2 = 0;
    der(err) = (x1-x1e)^2;
  end Roberts_opt;
end Roberts;
```

Listing 2: Calibrating Roberts model

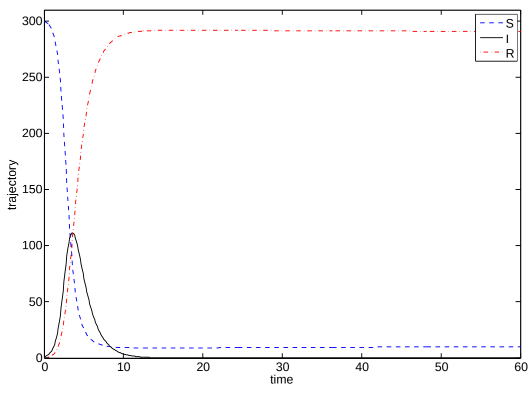

Figure 2: The calibration of the extended SIR model.

these parameters as the nominal model (its trajectories are denoted as xN in Figure 1). However, when the model was adjusted to the reference trajectory for x_1: $(0.0, 0.6)$, $(8e6, 0.3)$, $(16e6, 0.2)$, $(24e6, 0.1)$, $(32e6, 0.01)$, $(40e6, 0.0)$, $(4.0e7, 0.0)$ models parameters were changed to $p_1 = 1e-3$ (we assumed that lower bound for the parameter p_1 is $1e-3$), $p_2 = 2.1986e4$ and p_3 to $3.0e7$. Also trajectories of the model changed–they are shown on Fig. 1.

Example 2 The second example is related to the model of food borne diseases. The equations of the evolution of three populations: S (Susceptible population), I (Infected population), R (Recovered population) and the equation of the pathogen concentration evolution are given below.

$$\begin{aligned}\dot{S} &= -p_3(B/(B+p_2))S - ((p_4p_5)/p_8)SI - \\ &\quad p_1S + p_1p_8 \\ \dot{I} &= p_3(B/(B+p_2))S + ((p_4p_5)/p_8)SI - p_1I \\ \dot{R} &= p_1I - p_6R \\ \dot{B} &= p_7I - p_5B.\end{aligned}$$

The model has eight parameters, all of them are decision variables in our calibration problem. In addition we assume that initial pathogen concentration, $B(0)$, must be determined. The trajectories of the calibrated model are shown on Fig. 2.

8 Conclusions

The IDOS server is still a prototype. Although it is operational its widespread use can be hampered by the limited computing resources which are currently allocated to the server. Furthermore, adding a new solver to the server is not as simple as it should be. We plan to alleviate these problems in the near future.

References

[1] Åkesson J. Tools and Languages for Optimization of Large-Scale Systems. Lund, Sweden: *PhD thesis*, Department of Automatic Control, Lund University, 2007.

[2] Åkesson J. Optimica–an extension of Modelica supporting dynamic optimization. In: Proceedings of the 6th Modelica Conference 2008, Bielefeld, Germany, Modelica Association, 3-4 March 2008.

[3] Błaszczyk J, Karbowski A, K. Malinowski K. Object Object library of algorithms for dynamic optimization problems: benchmarking SQP and nonlinear interior point methods. In: Journal of Applied Mathematics and Computer Science, Vol. 17, 515–537, 2007.

[4] Bonami P, Lee J BONMIN Users' Manual, 2009.

[5] Bonami P, Lodi A, Cornujols G, Margot F. A feasibility pump for mixed integer nonlinear programs. In: Mathematical Programming, Vol. 119, 331–352, 2009.

[6] Franke R. Anwendung von Interior-Point-Methoden zur Lösung zeitdiskreter Optimalsteuerungsprobleme. *Master's thesis*, Techniche Universität Ilmenau, Institut für Automatisierungs- und Systemtechnik Fachgebiet Dynamik und Simulation ökologischer Systeme, Ilmenau, Germany, October 1994 (in German).

[7] Franke R. OMUSES — a Tool for the Optimization of MUltistage Systems and HQP — a Solver for Sparse Nonlinear Optimization. Dept. of Automation and Systems Engineering, Technical University of Ilmenau, Germany, September 1998.

[8] Gondzio J. Multiple centrality corrections in a primal-dual method for linear programming. *Technical Report 1994.20*, Department of Management Studies, University of Geneva, Geneva, Switzerland, 1994.

[9] Hairer E, Lubich Ch, Roche M. The numerical solution of differential–algebraic equations by Runge–Kutta methods. Lecture Notes in Mathematics 1409, Springer–Verlag, Berlin, Heidelberg, 1989.

[10] Hairer E, Norsett S.P, Wanner G. Solving Ordinary Differential Equations I., Springer–Verlag, Berlin Heidelberg New York, 2008.

[11] Hairer E, Wanner G. Solving Ordinary Differential Equations II, Springer–Verlag, Berlin Heidelberg New York, 1996.

[12] Hindmarsh, A.C., Brown, P.N., Grant, K.E., Lee, S.L., Serban, R., Shumaker, D.E., Woodward, C.S. SUNDIALS: Suite of Nonlinear and Differential/Algebraic Equation Solvers, preprint, UCRL-JRNL-200037, 2004.

[13] Mehrotra S. On the implementation of a primal–dual interior point method. In: SIAM Journal on Optimization, Vol. 2, 575–601, 1992.

[14] Oberle H.J, Grimm W. BNDSCO – A Program for the Numerical Solution of Optimal Control Problems, Report No. 515 der DFVLR (German Test and Research Institute for Aviation and Space Flight), 1989.

[15] Pantelides C. Consistent initialization of differential–algebraic system. In: SIAM J. Scientific and Statistical Computation Vol. 9, 213-231, 1988.

[16] Pryce, J.D. A simple structural analysis method for DAEs, BIT 41 (2001), pp. 364–394.

[17] Pytlak R. Numerical procedures for optimal control problems with state constraints. Lecture Notes in Mathematics 1707, Springer–Verlag, Heidelberg, 1999.

[18] Pytlak R. Numerical procedure for optimal control of higher index DAEs. In: J. Discrete and Continuous Dynamical Systems, Vol. 29, 1–24, 2011.

[19] Pytlak R, Tarnawski T, Fajdek B, Stachura M. Interactive dynamic optimization server–connecting one modeling language with many solvers. In: Optimization Methods and Software, 2013. `http://dx.doi.org/10.1080/10556788 .2013.799159`.

[20] Pytlak R (ed.). Interactive computer environment for solving optimal control problems–IDOS. BelStudio, Warsaw, Poland, 2012.

[21] Pytlak R, Błaszczyk J, Krawczyk K, Tarnawski T. Solvers chaining in the IDOS server for dynamic optimization. In: Proceedings of 52nd Conference on Decision and Control, Florence, Italy, 10-13 December 2013.

[22] Serban R, Hindmarsh A.C. CVODES, the sensitivity-enabled ODE solver in SUNDIALS. In: Proceedings of the 5th International Conference on Multibody Systems, Nonlinear Dynamics and Control, ASME, Long Beach, CA, 2005.

[23] de Souza, N., Fateman, R.J., Moses, J., Yapp, C. The Maxima Book, http://maxima.sourceforge.net/docs/ maximabook/maximabook-19-Sept-2004.pdf, 2004.

[24] Tarnawski, T, Pytlak R. DOML - a compiler environment for dynamic optimization supporting multiple solvers. Accepted for: 10th Modelica Conference 2014, Lund, Sweden, Modelica Association, 10-12 March 2014.

[25] Wachter A, Biegler L.T. On the implementation of an interior point Line search filter algorithm for large scale nonlinear programming. In: Math. Programming, Vol. 106, 25–57, 2006.

[26] Interactive Dynamic Optimization Server. `http://idos.mchtr.pw.edu.pl/`. November 2013.