Large-Scale Dynamic Optimization with the Directional Second Order Adjoint Method

Derya B. Özyurt and Paul I. Barton

Department of Chemical Engineering
Massachusetts Institute of Technology
66-464, 77 Massachusetts Avenue
Cambridge, MA 02139
derya@mit.edu, pib@mit.edu

Abstract

Efficient solution of large-scale dynamic optimization problems can be achieved by exploiting the advantages of state-of-the-art integration and large-scale nonlinear programming tools. Truncated-Newton method provides an effective way to solve large-scale optimization problems by achieving savings in computation and storage. For dynamic optimization, the Hessian-vector products required by these methods can be evaluated accurately at a computational cost which is usually insensitive to the number of optimization variables using a novel directional Second Order Adjoint (dSOA) method. The case studies presented in this paper demonstrate that a dSOA powered truncated-Newton method is a promising candidate for the solution of large-scale dynamic optimization problems.

Key words: directional second order derivatives, truncated-Newton method, staggered corrector method

Prepared for presentation at the 2004 Annual Meeting, Austin, TX, Nov. 7-12

1 Introduction

Dynamic optimization problems consist of an objective function(al) dependent on a set of state variables that are determined in general by differential-algebraic equations (DAEs). The efficient numerical solution of these problems is important in many applications where optimization of time-dependent performance is required, such as optimal control, parameter estimation, and dynamic system synthesis and analysis. Solution procedures for dynamic optimization problems often have to tackle with large numbers of state variables, optimization parameters, and constraints on the overall dynamic system, and/or those that are generated by the solution approach. An approach which decouples the optimization problem from the solution of the embedded dynamic system can enable exploitation of the full advantages of state-of-the-art integration and large-scale nonlinear programming tools.

If we consider the solution of optimal control problems as an example, then for a given instance of the parameter values characterizing the control discretization, the embedded DAE system is solved to provide the objective function value and its derivative values to be utilized by a gradient-based NLP solution procedure. Derivative information can be extracted from the first order sensitivities and there are several efficient methods to calculate these sensitivities[1, 2]. It has also been shown that for DAE embedded functionals the first order adjoint method can be more attractive when the number of parameters is large and there is one or a small number of functionals [3]. This is because, except in pathological situations [4], the cost of the adjoint calculations relative to the cost of a simulation is insensitive to the number of parameters involved, whereas for sensitivities this relative cost scales linearly with the number of parameters. For large-scale problems, if second order information is employed by the optimization procedure, directional second order derivatives are often estimated using directional finite differences based on a first order adjoint code. Specifically, the Hessian-vector product of the objective function

$$J(p) = g(x(t_f, p), p) + \int_{t_0}^{t_f} h(t, x(t, p), p)\, dt \tag{1}$$

at p^*in the direction u is estimated by

$$\nabla^2 J(p^*)u \approx \frac{\nabla J(p^* + \varepsilon u) - \nabla J(p^*)}{\varepsilon}, \tag{2}$$

which requires two state and adjoint integrations, one at p^* and one at $p^* + \varepsilon u$. The cost of computing this estimate relative to the cost of a simulation is insensitive to the number of parameters, by virtue of the properties of the adjoint integration.

Truncated-Newton methods are a class of optimization methods which employ directional second order information in updating the current estimate of the solution by approximately solving the Newton equations

$$\nabla^2 J(p^{(k)})u = -\nabla J(p^{(k)}) \tag{3}$$

using an iterative procedure [5, 6]. Each inner iteration of the truncated-Newton method requires evaluation of the Hessian-vector product, $\nabla^2 J(p^{(k)})u$, in a given direction u.

Incorporation of “exact” second order information within truncated-Newton methods has been undertaken previously for data assimilation problems [7, 8]. This rather specific case of the general directional second order adjoint method presented here, is implemented for non-stiff ODE embedded systems using a “direct” Automatic Differentiation (AD) [9] approach. Instead of applying AD to a code that integrates the embedded system, the dSOA method presented in this paper constructs differentiated subroutines to be integrated by an implicit integration scheme. This “targeted” AD approach reduces the computational cost and accurate Hessian-vector products are calculated in less time than two gradient evaluations [4]. In addition, except in pathological situations [4], the cost of the dSOA method relative to the cost of a simulation is insensitive to the number of parameters involved. Another approach to compute second order information for truncated-Newton methods is calculating the Hessian-vector products from directional second order forward sensitivities [10]. However, the cost of calculating the directional second order sensitivities, relative to the cost of a simulation, scales linearly with the number of parameters and for large-scale dynamic optimization problems these sensitivity calculations can become very expensive.

In this study, we assume that the dynamic optimization problem is given as

$$\min_{p} \quad J(p) = g(x(t_f, p), p) + \int_{t_0}^{t_f} h(t, x(t, p), p)\ dt \tag{4}$$

$$\dot{x} + F(t, x, p) = 0, \quad x(t_0) = x_0(p), \tag{5}$$

subject to the following bounds on the optimization parameters

$$p^L \leq p \leq p^U. \tag{6}$$

Although the following presentation only considers stiff ODE systems, by subtle consideration of initial conditions and stability of the adjoint systems, the theory and application can be extended to DAE embedded systems.

In the following section, a short description of Nash’s Truncated-Newton method [6] with dSOA as a directional second order derivative evaluator is given. Examples presented demonstrate the promise of the proposed approach.

2 “dSOA powered” truncated-Newton method

Truncated-Newton methods consist of an outer iteration for the nonlinear optimization problem and an inner iteration for approximate solution of the Newton equations. These methods provide an effective solution procedure for large-scale optimization problems if an efficient but approximate solution of the Newton equations can produce a “good” direction which can be used in the outer iteration with appropriate globalization strategies [5]. Techniques to calculate the requisite second order information efficiently and accurately can improve the overall method by expanding its applicability to large-scale dynamic optimization problems. The directional second

order adjoint method along with its first order counterpart computes the directional second order derivatives and gradients of the objective function efficiently and accurately, improving both the outer and inner iterations of the truncated Newton methods.

The truncated-Newton algorithm adapted to solve large-scale dynamic optimization problems can be stated as follows:

1. Set iteration number, $k = 0$ and specify an initial approximation $p^{(0)}$ to the optimal solution p^*.
2. If the approximation $p^{(k)}$ is a local minimizer of J within a given tolerance, Stop.
3. Solve Eqn. (3) approximately by a modified-Lanczos algorithm [6] using preconditioning [11] to obtain a search direction, u. At each iteration of this inner loop the Hessian-vector products required are calculated by the dSOA method.
4. Apply a line search to find $\alpha > 0$ such that $J(p^{(k)} + \alpha u) < J(p^{(k)})$. Gradient evaluations are performed by a first order adjoint method [3].
5. Set $p^{(k+1)} = p^{(k)} + \alpha u$, $k = k + 1$ and go to Step 2.

This algorithm is implemented using Nash's truncated-Newton code for optimization problems with bounds on variables [6]. The original code is modified to include the dSOA method in Step 3.

The evaluation of directional second order derivatives by the dSOA method requires solution of the state equations

$$\dot{x} + F(t, x, p) \;=\; 0, \quad x(t_0) = x_0(p),$$

and directional first order sensitivities

$$\dot{s} + F_x s + F_p u \;=\; 0, \quad s(t_0) = x_{0p} u,$$

forward in time. At the final time point of the forward integration (t_f), if the objective function consists of point-form functionals, initial values of the adjoint variables should be calculated. Then first order adjoint

$$\dot{\lambda}^T - \lambda^T F_x = -h_x, \;\; \lambda^T(t_f) = 0,$$

and directional second order adjoint equations

$$\dot{\mu} - F_x^T \mu = (\lambda^T \otimes I_{n_x})(F_{xp} u + F_{xx} s) - h_{xx} s - h_{xp} u, \;\; \mu(t_f) = 0,$$

must be integrated backward in time. Finally we can calculate the directional second order derivative of a integral functional $G(p)$ ($\equiv \int_{t_0}^{t_f} h(t, x(t,p), p)$) by

$$\begin{aligned} \frac{\partial^2 G}{\partial p^2}\, u \;=\; & \int_{t_0}^{t_f} \{h_{pp}\, u + h_{px}\, (x_p u) - \left[F_p^T \mu + (\lambda^T \otimes I_{n_p})(F_{pp}\, u + F_{px}\, (x_p u))\right]\} dt \\ & + \left[(\lambda^T \otimes I_{n_p}) x_{pp}\, u + x_p^T \mu\right]_{t=t_0}. \end{aligned} \tag{7}$$

The size of the first four systems is independent of n_p, the number of parameters. On the other hand, the formulation requires evaluation of several vector-matrix, matrix-vector and vector-matrix-vector products. Therefore a successful implementation of the dSOA method is achieved by calculating these terms accurately and efficiently using AD [9]. The details of the dSOA method can be found in [4].

Estimating the Hessian-vector product using directional finite differences (Eqn. 2) requires two gradient evaluations at the first iteration of the first inner loop (one at $p^{(0)}$ and one at $p^{(0)} + \varepsilon u$) of the truncated-Newton method. However, at the second and subsequent iterations of each inner loop only a single gradient evaluation is required, at $p^{(k)} + \varepsilon u$, because $p^{(k)}$ remains fixed and only the direction u changes. Our numerical experience shows that evaluation of the directional second order derivatives using dSOA will be roughly 10-30% more expensive than a single gradient evaluation using the first order adjoint method [4]. This means that the cost of inner iterations for the "dSOA powered" truncated-Newton method will be only marginally different from estimation using directional finite differences. The benefit of our proposed method, therefore, will be seen empirically in the overall time for optimization because of the higher accuracy of the second order information resulting in faster and/or more robust convergence. Moreover, success in approximating the second order derivatives via gradients requires a proper ε value for the finite difference method. Elimination of this potentially failure prone or sometimes impossible selection is another advantage of the "dSOA powered" truncated-Newton method.

3 Examples

The following case studies showcase the effectiveness and efficiency of the directional second order adjoint method in conjunction with a well known truncated-Newton method. Numerical experiments are performed on a Pentium IV/3.20 GHz Shuttle X machine with 1 GB memory and running Linux kernel 2.4. All automatic differentiation tasks are performed by the AD tool TAMC [12].

3.1 Canned food sterilization

The problem of industrial sterilization [10, 13] of canned food is considered as the first example. This involves modeling heating and thermal degradation processes. Heating of the canned foods by conduction is modeled by employing Fourier's second law on cylindrical coordinates. The PDE is reduced into an ODE using the numerical method of lines. In addition, the thermal degradation of the microorganisms and nutrients are assumed to obey pseudo first order kinetics. We consider the final retention of a nutrient within total volume of the container as our functional.

Similar to the procedure described in [10] an approximate solution is obtained for the above stated problem by assuming both inequality constraints are active at the optimal solution

and incorporating them as quadratic penalty terms within the objective function. The integration tolerance is 10^{-7} and the optimization accuracy (oa) value is set to 10^{-5}. In this example with 302 equations, only restarting of the integration at the control parameter grid points contributes to the increase of the computational cost per iteration. For an initial guess, $T_{retort} = 100.0$ (except for $n_p = 56, 112$, where the last 5 and 12 constant profile value guesses are set to 20.0) and increasing the number of control parameterization grid points (n_p), computational results are listed in Table 1. Computing directional second order derivatives with dSOA is advantageous considering that its average time per iteration increases less than twice for doubled number of parameters. Another reason for employing dSOA for the evaluation of second order derivative

Table 1: Computational results for the canned food sterilization example (n_p: number of parameters, NIT: number of iterations, NF: number of outer iterations, CG: number of inner iterations, J: objective function value, CPU: overall CPU time, TPI: average CPU time per iterations).

n_p	NIT	NF	CG	J	$CPU(s)$	$TPI(s)$
7	17	20	41	0.4678	172.4	2.8
14	25	28	60	0.4711	387.6	4.4
28	45	54	133	0.4721	1382.4	7.4
56	46	59	112	0.4722	2385.1	14.0
112	97	110	248	0.4717	9073.1	25.3

information is that the finite difference approximation to estimate the same second order information often fails to give sufficiently accurate directions for the optimization and results in termination of the optimization procedure without finding a solution. In all cases the gradient evaluation at the perturbed optimization parameter values resulted in physically unreasonable state values (e.g., negative temperatures) and therefore erroneous gradients, consequently failing the outer iterations. It was not possible to solve this problem using directional finite differences.

On the other hand, the directional second order sensitivity based approach can solve the canned food sterilization problem at a higher cost (Table 2). Collectively, the average CPU

Table 2: Computational results for the canned food sterilization example using directional second order forward sensitivities.

n_p	NIT	NF	CG	J	$CPU(s)$	$TPI(s)$
7	17	20	40	0.4679	342.9	6.9
14	22	25	48	0.4702	976.0	16.1
28	37	51	105	0.4712	5698.5	43.7

time per iteration for the “dSOA powered” truncated-Newton method is very comparable with the same of the FDM based method (Figure 1). Here TPI for FDM is obtained by averaging the time elapsed per iteration before any failure occurs.

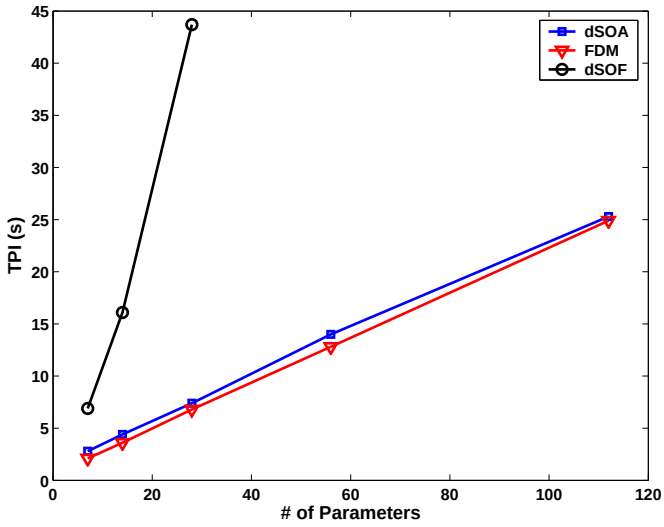

Figure 1: Comparison of TPI for dSOA, FDM and dSOF

3.2 Temperature profile matching

A two dimensional boundary control heating problem adopted from [14] is considered as the second case study. A rectangular domain is heated by controlling the temperature on its boundaries so that a prespecified temperature-time trajectory is approximately followed within a specified interior subdomain. The nonlinear parabolic PDE is reduced into an ODE using the numerical method of lines [14].

For an integration tolerance at 10^{-10} and the optimization accuracy value of 10^{-8}, several cases with a gradually increasing number of parameters were tested. The number of parameters is increased both by finer control parameterization and addition of initial conditions as degrees of freedom to the optimization. These latter additions are noted by a $(+)$ sign in Column 2 of Table 3, i.e., $10 + 450$ denotes 10 control parameters and 450 initial conditions are considered as optimization parameters. Nonlinearity is introduced by the addition of a nonlinear heat source term and noted by (nl) in Column 2.

From the last Column of Table 3 we conclude that the computational cost per iteration of a "dSOA powered" truncated-Newton method stays constant when more parameters are added to the optimization problem. The only increase is caused by integration restarts at each control vector parameterization time point. Obviously, the total cost of optimization increases because the number of iterations increases.

Finite difference approximation to the directional second order derivatives does not result in a robust performance (Table 4). Especially, for cases with larger number of parameters or nonlinearity, the dSOA based method outperforms this approach in terms of attained objective function value at a comparable overall computational cost.

Table 3: Computational results for the temperature matching example (n_x: number of equations, n_p: number of parameters, NIT: number of iterations, NF: number of outer iterations, CG: number of inner iterations, J: objective function value, CPU: overall CPU time, TPI: average CPU time per iterations).

n_x	n_p	NIT	NF	CG	$J(\times 10^5)$	$CPU(s)$	$TPI(s)$
45	40	33	34	105	4.295	69.3	0.5
	40+5	30	30	100	4.325	65.4	0.5
	40+20	34	35	101	4.582	67.4	0.5
	160	78	79	206	4.812	464.1	1.6
	320	128	129	269	7.779	1108.8	2.8
	320(nl)	115	120	229	75.99	983.4	2.8
861	10+5	12	13	36	6.922	1049.6	21.4
	10+450	32	33	183	6.258	5079.1	23.5
	20	22	23	80	4.736	4019.5	39.0
	50	48	49	171	4.193	17660	80.3

4 Conclusions

An efficient method to calculate accurate directional second order derivatives is incorporated into the truncated-Newton method to solve large-scale dynamic optimization problems. Since dSOA has only "weak" n_p-dependence, as the number of parameters is increased, the relative cost of derivative evaluations for large dynamic systems with many parameters does not amplify, staying constant if no integration restarts are necessary.

Although the computational costs of evaluating a gradient and a directional second order derivative with dSOA are comparable for a large number of parameters, obtaining the latter accurately improves the computation time by reducing the total number of iterations. Therefore, the directional finite difference method using gradient evaluations can give similar optimization results with comparable computational cost but with decreased accuracy provided that there are no numerical difficulties using the procedure.

The existing implementation can be improved in several ways. Since an additional direction can be obtained even "cheaper" [4], linear biconjugate methods can be used for the inner iterations to solve the Newton equations (3). The dSOA method to calculate directional second order derivatives of ODE embedded functionals can be extended to include differential-algebraic equations. Moreover, for larger systems constructed by PDE semi-discretizations iterative linear solvers can be utilized to reduce the cost of integration.

Table 4: Computational results for the temperature matching example using finite difference approximation of the Hessian-vector product (* marks the runs where the optimization procedure terminated because no significant improvement in the objective function value is achieved).

n_x	n_p	NIT	NF	CG	$J(\times 10^5)$	$CPU(s)$	$TPI(s)$
45	40	33	33	121	4.330	61.1	0.4
	40+5	30	31	94	4.358	49.0	0.4
	40+20	24	25	54	11.95^*	31.5	0.4
	160	83	84	222	4.600	412.8	1.4
	320	112	113	224	10.80^*	883.4	2.6
	320(nl)	110	114	219	155.55^*	899.7	2.7
861	10+5	13	14	39	6.922	988.1	18.6
	10+450	7	8	16	16.86^*	444.9	18.5
	20	24	25	80	4.735	3506.1	33.4
	50	27	28	83	4.606^*	8116.8	73.1

References

[1] W. F. Feehery, J. E. Tolsma, and P. I. Barton. Efficient sensitivity analysis of large-scale differential-algebraic systems. *Applied Numerical Mathematics*, 25:41–54, 1997.

[2] Timothy Maly and Linda Petzold. Numerical methods and software for sensitivity analysis of differential algebraic equations. *Applied Numerical Mathematics*, 20:57–79, 1996.

[3] Y. Cao, S. Li, L. Petzold, and R. Serban. Adjoint sensitivity analysis for differential-algebraic equations: The adjoint DAE system and its numerical solution. *SIAM Journal on Scientific Computing*, 24(3):1076–1089, 2003.

[4] Derya B. Özyurt and Paul I. Barton. Cheap second order directional derivatives of stiff ODE embedded functionals. In press: *SIAM Journal on Scientific Computing*, 2004.

[5] Stephen G. Nash. A survey of Truncated-Newton methods. *Journal of Computational and Applied Mathematics*, 124:45–59, 2000.

[6] Stephen G. Nash. Newton-type minimization via the Lanczos method. *SIAM Journal on Numerical Analysis*, 21(4):770–788, 1984.

[7] Francois-Xavier Le Dimet, I. M. Navon, and D. N. Daescu. Second-order information for data assimilation. *Monthly Weather Review*, 130:629–648, 2002.

[8] Z. Wang, I. M. Navon, X. Zou, and F. X. Le Dimet. A truncated Newton optimization algorithm in meteorology applications with analytic Hessian/vector products. *Computational Optimization and Applications*, 4:241–262, 1995.

[9] Andreas Griewank. *Evaluating derivatives: Principles and techniques of algorithmic differentiation*. SIAM, Philadelphia, 2000.

[10] E. B. Canto, J. R. Banga, A. A. Alonso, and V. S. Vassiliadis. Restricted second order information for the solution of optimal control problems using control vector parameterization. *Journal of Process Control*, 12:243–255, 2002.

[11] Stephen G. Nash. Preconditioning of Truncated-Newton methods. *SIAM Journal on Scientific and Statistical Computing*, 6(3):559–616, 1985.

[12] Ralf Giering. *Tangent linear and Adjoint Model Compiler: Users Manual 1.4*. http://www.autodiff.com/tamc, 1999.

[13] J. R. Banga, R. I. Perez-Martin, J. M. Gallardo, and J. J. Casares. Optimization of the thermal processing of conduction-heated canned foods: Study of several objective functions. *Journal of Food Engineering*, 14:25–51, 1991.

[14] Linda Petzold, J. B. Rosen, P. E. Gill, L. O. Jay, and Park K. Numerical optimal control of parabolic PDEs using DASOPT. In L. T. Biegler, T. F. Coleman, A. R. Conn, and F. N. Santosa, editors, *Large Scale Optimization with Applications: Part II*. Springer-Verlag, New York, 1997.