

Cheap Second Order Directional Derivatives of Stiff ODE Embedded Functionals

Derya B. Özyurt and Paul I. Barton

Department of Chemical Engineering
Process Systems Engineering Laboratory
Massachusetts Institute of Technology

Wednesday, 19th November 2003

Derivatives of Stiff ODE Embedded Functionals

$$g(x(t_f, p), p),$$
$$G(p) = \int_{t_0}^{t_f} g(x(t, p), p) dt$$

where $x(t, p)$ is defined by

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

First order derivatives: $\frac{\partial g}{\partial p}, \frac{\partial G}{\partial p}$

Second order derivatives: $\frac{\partial^2 g}{\partial p^2}, \frac{\partial^2 G}{\partial p^2}$

Directional derivatives: $\frac{\partial^2 g}{\partial p^2} u, \frac{\partial^2 G}{\partial p^2} u$

Algorithmic Differentiation - Motivation

Main results for the evaluation of gradients and second order adjoints ¹ using reverse mode and forward over reverse mode, respectively.

$$\mathit{cost}\{\mathit{grad}(F)\} \approx 4 \times \mathit{cost}\{\mathit{eval}(F)\}$$

$$\mathit{cost}\{\mathit{soad}(F)\} \approx 10 \times \mathit{cost}\{\mathit{eval}(F)\}$$

¹A. Griewank. *Evaluating derivatives: Principles and techniques of algorithmic differentiation*. SIAM, Philadelphia, 2000

Why not “direct AD”?

- Non-stiff ODE system with an explicit solver: “direct AD”
 - Stiff ODE or DAE
 - solvers are more complex; corrector iteration \Rightarrow Applying AD directly is not reasonable
 - Exploiting the structure of the state and sensitivity system (e.g. staggered corrector method ²)
- \Rightarrow “targeted AD”

²W. 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

How are the derivatives calculated?

- First order forward sensitivities ³

Sensitivity Equations:

$$\frac{\partial \dot{x}}{\partial p_i} + \frac{\partial F}{\partial x} \frac{\partial x}{\partial p_i} + \frac{\partial F}{\partial p_i} = 0, \quad \frac{\partial x(t_0)}{\partial p_i} = \frac{\partial x_0}{\partial p_i}, \quad i = 1, \dots, n_p.$$

Finally,

$$\frac{\partial G}{\partial p} = \int_{t_0}^{t_f} (g_p + g_x x_p) dt$$

- First order adjoint method ⁴

³W. 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

⁴Y. 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 of Scientific Computing*, 24(3):1076–1089, 2003

How are they calculated?

- Second order forward sensitivities ⁵
Sensitivity Equations

$$\begin{aligned} & \frac{\partial^2 \dot{x}}{\partial p^2} + \left[\frac{\partial F}{\partial x} \otimes I_{n_p} \right] \frac{\partial^2 x}{\partial p^2} \\ & + \left[I_{n_x} \otimes \left(\frac{\partial x}{\partial p} \right)^T \right] \left[\frac{\partial^2 F}{\partial x^2} \frac{\partial x}{\partial p} + \frac{\partial^2 F}{\partial p \partial x} \right] \\ & + \left[\frac{\partial^2 F}{\partial x \partial p} \frac{\partial x}{\partial p} + \frac{\partial^2 F}{\partial p^2} \right] = 0, \quad \frac{\partial^2 x(t_0)}{\partial p^2} = \frac{\partial^2 x_0}{\partial p^2} \end{aligned}$$

- Second order adjoint method ⁶: “direct AD”

⁵V. S. Vassiliadis, E. B. Canto, and J. R. Banga. Second-order sensitivities of general dynamic systems with application to optimal control problems. *Chemical Engineering Science*, 54:3851–3860, 1999

⁶Z. 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

Derivation of Directional Derivatives

State equations

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

Directional first order sensitivities

$$\dot{x}_p u + F_x(x_p u) + F_p u = 0, \quad x_p u|_{t=0} = x_p(t_0)u$$

First order adjoint system

$$\begin{aligned} \dot{\lambda} - F_x^T \lambda &= -g_x^T \\ \lambda|_{t=t_f} &= 0 \end{aligned}$$

Derivation of Directional Derivatives

Directional second order adjoint system

$$\begin{aligned}\dot{\lambda}_p u - F_x^T \lambda_p u &= (\lambda^T \otimes I_{n_x})(F_{xp} u + F_{xx}(x_p u)) \\ &\quad - g_{xx}(x_p u) - g_{xp} u \\ (\lambda_p u)|_{t=t_f} &= 0\end{aligned}$$

Second order directional derivatives

$$\begin{aligned}\frac{\partial^2 G}{\partial p^2} u &= \int_{t_0}^{t_f} \{g_{pp} u + g_{px}(x_p u) \\ &\quad - [F_p^T(\lambda_p u) + (\lambda^T \otimes I_{n_p})(F_{pp} u + F_{px}(x_p u))]\} dt \\ &\quad + [(\lambda^T \otimes I_{n_p})x_{pp} u + x_p^T(\lambda_p u)]|_{t=t_0}.\end{aligned}$$

AD usage

- Algorithmic differentiation (AD) is used to construct the system to be solved efficiently and without truncation errors.
- Second Order Adjoint: **FOA**
- Complexity results for gradient and Hessian-vector products
- “Differentiate then discretize”
- Mostly used TAMC ⁷

⁷R. Giering. *Tangent linear and Adjoint Model Compiler: Users Manual 1.4*.
<http://www.autodiff.com/tamc>, 1999

AD

We need to evaluate

$$\begin{aligned} & F_x x_p u + F_p u \\ & \lambda^T F_x \\ & g_x \\ & F_x^T (\lambda_p u) \\ & (\lambda^T \otimes I_{n_x}) (F_{x_p} u + F_{x x} (x_p u)) \\ & g_{x x} (x_p u) - g_{x p} u \\ & g_{p p} u + g_{p x} (x_p u) \\ & F_p^T (\lambda_p u) + (\lambda^T \otimes I_{n_p}) (F_{p p} u + F_{p x} (x_p u)) \\ & (\lambda^T \otimes I_{n_p}) x_{p p} u + x_p^T (\lambda_p u) \end{aligned}$$

Also, for the integrator the full Jacobian of F , i.e. F_x , is needed.

Numerical Procedure

- Problem to be solved:
 - State equations, n_x
 - Directional forward sensitivity equations, n_x
 - First order adjoint equations, n_x
 - Directional second order adjoint equations (Sensitivities of the first order adjoints), n_x
- Needed:
 - Adjoint Equations : AD
 - Construction of the Quadratures: AD
 - Discretization and Integration/Sensitivity Calculation: IVP/Sensitivity solver with forward and backward integration capability

Implementation Overview

- DASPADJOINT : First Order Adjoint Sensitivity Solver ⁸
- Modifications to perform consecutively
 - Solve state equations (forward in time) along with first order directional sensitivities: staggered corrector method ⁹
 - Calculate initial values for both adjoint systems at t_f
 - Solve first order adjoint equations (backward in time) along with second order directional adjoint equations: staggered corrector method + Quadrature variables

⁸S. Li and L. Petzold. Description of DASPADJOINT: An adjoint sensitivity solver for differential-algebraic equations. Technical report, University of California, Department of Computer Science, Santa Barbara, CA 93106, 2001

⁹W. 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

Finite Difference Method (FDM) using First Order Adjoint System

Directional second order derivatives are often estimated using directional finite differences based on a first order adjoint code:

$$\left. \frac{\partial^2 G}{\partial p^2} \right|_{p=p^*} u \approx \frac{\nabla G(p^* + \varepsilon u) - \nabla G(p^*)}{\varepsilon}$$

which requires two state and adjoint integrations, one at p^* and one at $p^* + \varepsilon u$.

Example: Heat Equation

$$G(z(t_f, p), p) = \int_{t_0}^{t_f} \sum_{i=1}^{NEQ} z_i dt$$

$$z_t = p_1 z_{xx} + p_2 z_{yy}$$

Initial conditions:

$$z_{t=0} = 16x(1-x)y(1-y)$$

where $t_0 = 0$ and $t_f = 0.16$.

Example: Heat Equation (cont'd)

Computational Results

n_x ($M \times M$)	n_p	N_{JRE}	N_{steps}	N_{RES}	N_{SRES}
1600	1762	100	576	10553	664
6400	6722	112	589	20585	658
10000	10402	113	617	25415	747
22500	23102	112	656	36475	765

Comparisons with Simulation, Finite Difference Method using First Order Adjoint System and second direction

n_x ($M \times M$)	n_p	$cost(SIM)$ (CPU s.)	$\frac{cost(dSOA)}{cost(SIM)}$	$\frac{cost(dSOA)}{cost(FDM)}$	$\frac{cost(dSOA_2)}{cost(dSOA_1)}$
1600	1762	5.83	3.7	0.79	1.31
6400	6722	70.55	3.0	0.70	1.24
10000	10402	179.53	2.7	0.67	1.30
22500	23102	932.84	3.0	0.62	1.13

Conclusions

For a general ODE system the formulation to compute cheap second order directional derivative information is described.

- “differentiate then discretize” strategy
- The method is implemented by modification of DASPKADJOINT
- Several problems are solved with different structure and size
- In general: directional Second Order Adjoint method (dSOA) cost is approaching to approx. 50-60% of the FDM cost
- Subsequent directions are even “cheaper”
- Promising: Iterative methods - directional information (Hessian-vector products)
 - conjugate gradient
 - biconjugate methods even better
 - Arnoldi methods

Future Directions

- DAE implementation
- Solution of the optimal control problems via Truncated Newton method
- Efficient AD utilization (Second order AD, compact problem generation)
- Application to hybrid systems

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 0205590.

Notation

If \mathbf{A} is a $r \times s$ and \mathbf{B} a $t \times u$ matrix then their Kronecker product, $\mathbf{A} \otimes \mathbf{B}$ (a $r \cdot t \times s \cdot u$ matrix), is formed by replacing each a_{ij} in \mathbf{A} by $a_{ij}\mathbf{B}$. The derivative of matrix products is given as

$$\frac{\partial}{\partial x} \mathbf{A} \mathbf{C} = (\mathbf{A} \otimes \mathbf{I}_t) \frac{\partial \mathbf{C}}{\partial x} + (\mathbf{I}_r \otimes \mathbf{C}^T) \frac{\partial \mathbf{A}}{\partial x}$$

where \mathbf{A} is a $r \times s$ and \mathbf{C} a $s \times t$ matrix.

$$F = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_{n_x-1} \\ f_{n_x} \end{bmatrix} \quad F_p \equiv \frac{\partial F}{\partial p} = \begin{bmatrix} \frac{\partial f_1}{\partial p_1} & \cdots & \frac{\partial f_1}{\partial p_{n_p}} \\ \vdots & \vdots & \vdots \\ \frac{\partial f_{n_x}}{\partial p_1} & \cdots & \frac{\partial f_{n_x}}{\partial p_{n_p}} \end{bmatrix}$$

Notation (cont'd)

$$F_{px} \equiv \frac{\partial^2 F}{\partial x \partial p} = \begin{bmatrix} \frac{\partial^2 f_1}{\partial x_1 \partial p_1} & \cdots & \frac{\partial^2 f_1}{\partial x_{n_x} \partial p_1} \\ \vdots & \vdots & \vdots \\ \frac{\partial^2 f_1}{\partial x_1 \partial p_{n_p}} & \cdots & \frac{\partial^2 f_1}{\partial x_{n_x} \partial p_{n_p}} \\ \vdots & \vdots & \vdots \\ \frac{\partial^2 f_{n_x}}{\partial x_1 \partial p_1} & \cdots & \frac{\partial^2 f_{n_x}}{\partial x_{n_x} \partial p_1} \\ \vdots & \vdots & \vdots \\ \frac{\partial^2 f_{n_x}}{\partial x_1 \partial p_{n_p}} & \cdots & \frac{\partial^2 f_{n_x}}{\partial x_{n_x} \partial p_{n_p}} \end{bmatrix}$$

$$\frac{\partial F_x}{\partial p} \equiv F_{xx} x_p + F_{xp}$$