

Practical Target Applications of OpenAD in Chemical Engineering

Derya B. Özyurt and Paul I. Barton

Department of Chemical Engineering
Massachusetts Institute of Technology

Thursday, 20th January 2005

Overview

- A. Overview
- B. Current Status
- C. OpenAD - ChE Applications: Preparation
- D. OpenAD - ChE Applications: Experience
- E. dSOA-OpenAD

Update

- directional Second Order Adjoint (dSOA) method for stiff ODE embedded functionals
- dSOA for large-scale dynamic optimization
- dSOA for differential-algebraic equations (DAE) embedded functionals

dSOA for stiff ODEs

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

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

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

Directional second order derivatives: $\frac{\partial^2 h}{\partial p^2} u, \frac{\partial^2 G}{\partial p^2} u$

- parameter interactions, uncertainty estimation, second order optimization methods

dSOA

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

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

dSOA

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

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

dSOA-ODE: Summary

- “differentiate then discretize” strategy
- The method is implemented by modification of DASPKADJOINT
- Several problems are solved with different structure and size
- In general: dSOA cost is approaching to approx. 50-60% of the FDM cost
- Subsequent directions are even “cheaper”

dSOA and large-scale dynamic optimization

Dynamic optimization (DO) problems have non-sparse Hessians probably ill-conditioned.

Truncated Newton (TN) methods can use Hessian-vector information.

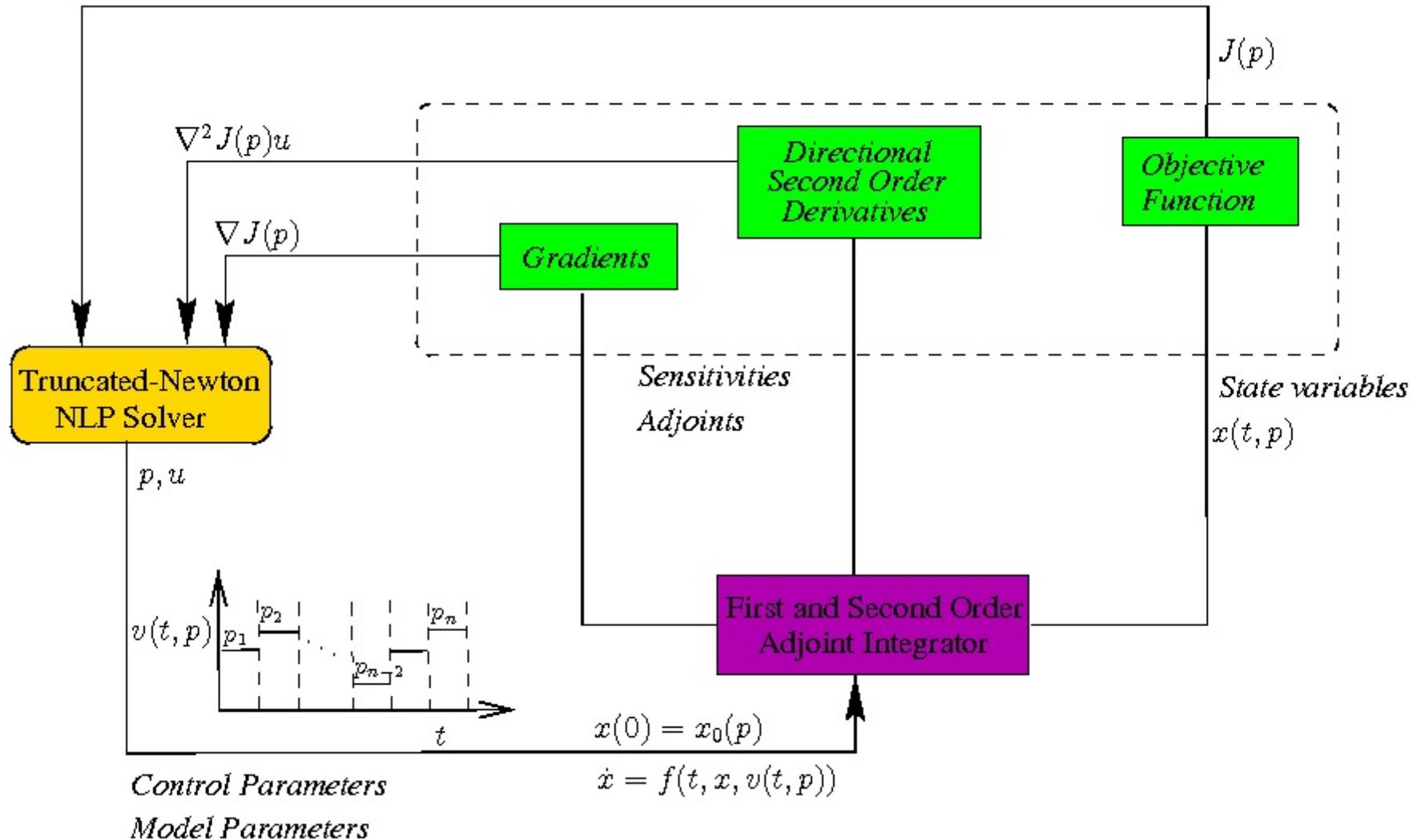
dSOA: an efficient and accurate Hessian-vector product evaluator.

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

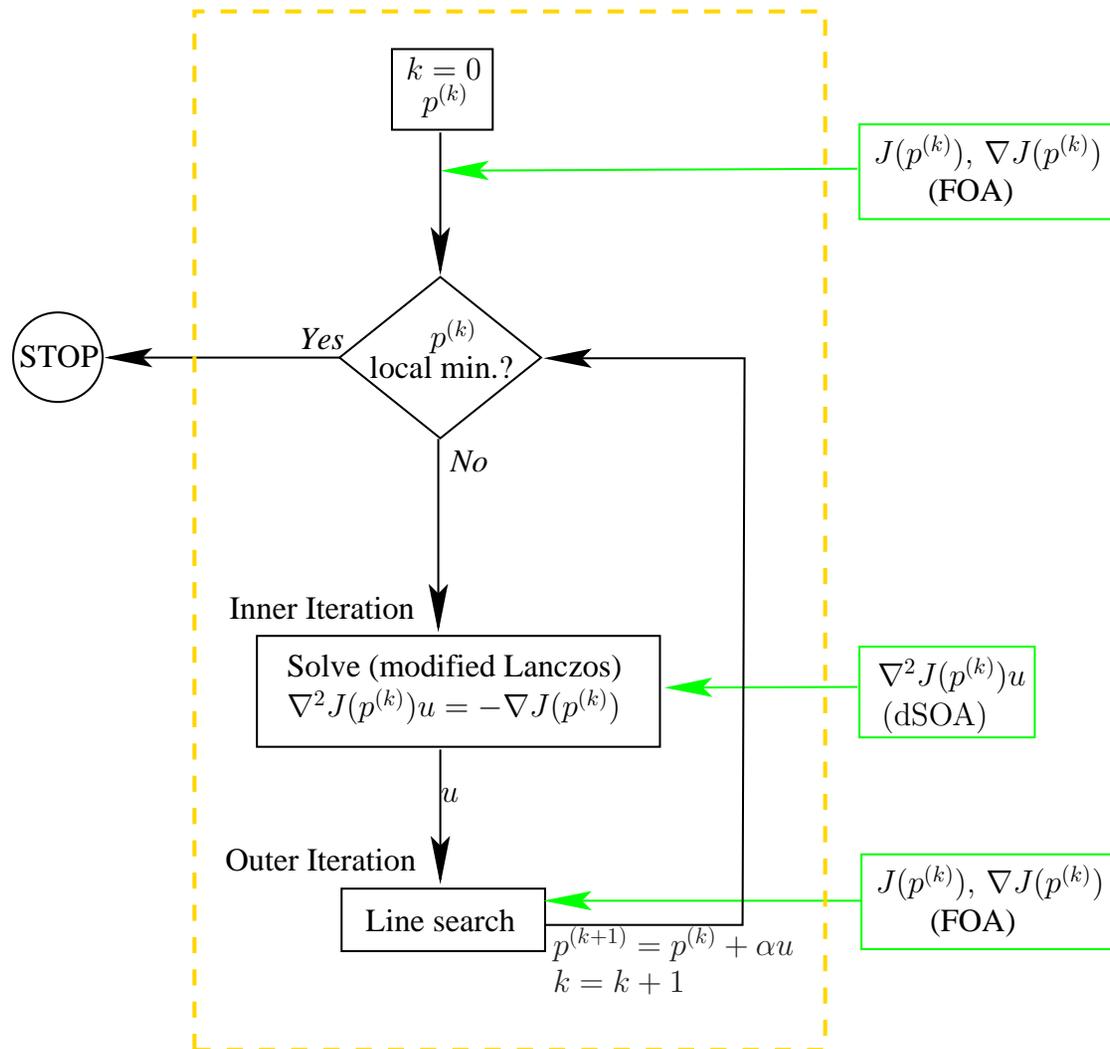
subject to

$$\begin{aligned} f(t, x, \dot{x}, p) &= 0 \\ f_0(t_0, x(t_0), \dot{x}(t_0), p) &= 0 \\ p^L &\leq p \leq p^U, \end{aligned}$$

Vector parameterization based solution procedure for DO



“dSOA powered” TN



dSOA-DO: Summary

- Incorporation of dSOA to calculate accurate directional second order derivatives effectively into TN to solve large-scale DO problems
- dSOA-TN increases the robustness of the TN and potentially improves the optimization time by reducing the total number of iterations
- Improvements:
 - Better interaction of dSOA and TN
 - Efficient code construction with AD
 - Two directional methods
 - Incorporation of the (in)equality constraints

dSOA and DAE

Consider a general parameter dependent index-1 DAE system

$$\begin{aligned} F(t, \dot{x}, x, p) &= 0, \\ x(t_0) &= x_0(p) \end{aligned}$$

where $x \in \mathbb{R}^{n_x}$ and $p \in \mathbb{R}^{n_p}$. Its corresponding first order sensitivity equations can be stated as

$$\begin{aligned} F_{\dot{x}} \dot{x}_p + F_x x_p + F_p &= 0, \\ x_p(t_0) &= x_{p_0}(p). \end{aligned}$$

Proposition: Suppose that $F_{\dot{x}}$ has constant rank. Then the differentiation index of the DAE is 1 if and only if its local index is 1. The local index is defined as the index of the pencil $cF_{\dot{x}}(\hat{t}, \hat{\dot{x}}, \hat{x}, \hat{p}) + F_x(\hat{t}, \hat{\dot{x}}, \hat{x}, \hat{p})$.

First order sensitivity equations for a general index-1 DAE system are linear time varying index-1 DAEs.

First Order Adjoint Equations

Now using any factorization for $F_{\dot{x}} = A(t, p) D(t, p)$ with *well matched* $A(t, p)$ and $D(t, p)$, we can write sensitivity equations as

$$\begin{aligned} A(t, p) (D(t, p)x_p(t))' + B(t, p) x_p(t) + F_p(t, p) &= 0 \\ D(t_0, p)x_p(t_0) &\in \text{im } D(t_0, p), \end{aligned}$$

where $B(t, p) = F_x(t, p) - A(t, p) D'(t, p)$. The initial condition $D(t_0, p)x_p(t_0) \in \text{im } D(t_0, p)$ can be replaced by

$$D(t_0, p)(x_p(t_0) - x_{p_0}(p)) = 0, \quad x_{p_0}(p) \in \mathbb{R}^{n_x}.$$

The derivative of an integral-form functional with respect to parameters, p ,

$$\frac{\partial G}{\partial p} = \int_{t_0}^{t_f} (g_p - \lambda^T F_p) dt - \int_{t_0}^{t_f} (-g_x + \lambda^T B - (\lambda^T A)' D) x_p dt - (\lambda^T A D x_p) \Big|_{t=t_0}^{t=t_f}.$$

First Order Adjoint Equations

By the inclusion of the first order adjoint equations

$$\begin{aligned}D^T(A^T \lambda)' - B^T \lambda &= -g_x^T, \\ A^T \lambda|_{t=t_f} &= 0,\end{aligned}$$

the first order derivatives of an integral-form functional become

$$\frac{\partial G}{\partial p} = \int_{t_0}^{t_f} (g_p - \lambda^T F_p) dt + (\lambda^T A D x_p)|_{t=t_0}.$$

Second Order Equations

To obtain the second order derivatives with respect to the parameters, we simply differentiate

$$\begin{aligned}
 \frac{\partial^2 G}{\partial p^2} &= \int_{t_0}^{t_f} \left\{ g_{pp} + g_{px} x_p - \left[F_p^T \lambda_p + (\lambda^T \otimes I_{n_p})(F_{pp} + F_{px} x_p + F_{p\dot{x}} \dot{x}_p) \right] \right\} dt \\
 &+ \left\{ [(\lambda^T AD) \otimes I_{n_p}] x_{pp} + x_p^T [(AD)^T \lambda_p \right. \\
 &+ \left. (I_{n_x} \otimes \lambda^T)((AD)^T)_p + ((AD)^T)_x x_p + ((AD)^T)_{\dot{x}} \dot{x}_p \right\} \Big|_{t=t_0}.
 \end{aligned}$$

Here, the equations for the second order adjoints, λ_p , are derived by first differentiating the first order adjoint equations:

$$\begin{aligned}
 D^T(A^T \lambda)'_p &+ [I_{n_x} \otimes (\lambda^T A)'] [(D^T)_{\dot{x}} \dot{x}_p + (D^T)_x x_p + (D^T)_p] \\
 &- B^T \lambda_p - (I_{n_x} \otimes \lambda^T)(B^T)_p \\
 &= -g_{xp}^T - g_{xx}^T x_p \\
 (A^T \lambda)_p|_{t=t_f} &= 0.
 \end{aligned}$$

dSOA-DAE: Summary

- Challenge: Factorization of $F_{\dot{x}}$ and obtaining generalized reflexive inverses.
- Challenge: The complicating terms can be eliminated by augmented formulation
- Challenge: Numerical solution procedure. Modification of the k -step BDF applied to the DAE can be necessary:

$$A(t_n) \frac{1}{h} \sum_{j=0}^k \alpha_j D(t_{n-j}) x_{n-j} + B(t_n) x_n = q(t_n).$$

- Challenge: Reformulation when not numerically qualified.

dSOA - AD process

A list of derivatives required by dSOA

First Order:	$\tilde{h}_x, \tilde{h}_p, \mu^T F_p (\lambda^T F_x), F_x s, F_p u, \lambda^T F_x (\mu^T F_x)$
Second Order:	$\tilde{h}_{xx} s + \tilde{h}_{xp} u, \tilde{h}_{px} s + \tilde{h}_{pp} u,$ $(\lambda^T \otimes I_{n_p})(F_{pp} u + F_{px} s),$ $(\lambda^T \otimes I_{n_x})(F_{xp} u + F_{xx} s)$

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)
- \Rightarrow “targeted AD”

Automatic Fortran Code generation for ChE

Application examples

- For several simulation/parameter estimation examples FORTRAN 77 codes are automatically generated by Jacobian software.

Gas and Oil (5.7 KB), Bubble Point Calculation (3.9 KB), Ethyl Acetate Reactor (25 KB), Ethyl Acetate Reactor (Variation 1) (25 KB), Lorenz Equations (3.4 KB), Orthogonal Collocation Example (4.4 KB), Phase Shift Oscillator (9 KB), Series Reactions (3.8 KB), Dynamic Estimation Example (3.1 KB), Simple Parameter Estimation Example (3.3 KB), EGF Activation Model (9.6 KB), A Continuous Distillation Column (105 KB), A Batch Rectifier (55 KB), A Simple Reactor Simulation (5.8 KB), Another Reactor Simulation (13 KB), A Binary Distillation Example (49 KB)

- Dependent variables: f
Independent variables: \dot{x} , x , $rpar$
Single file with several subroutines

The main subroutine

```
subroutine m_gasoil(ne,f,ni,g,t,x,xdot,ix,pix,isel,is,pis,
+ ipar,ii,pii,rpar,ir,pir,iu1,iu2)
implicit double precision(d)
implicit integer(i-n)
integer ne,ni,ipar(*),isel(*)
...
common /validate/ lvalidate
parameter(maxlocals=100)
double precision dlocal(maxlocals)
! set unit pointers
l_u=1
! call procedures for subunits
pis0=pis
...
! >>> procedure call for unit u
pis=pis+1
...
call m_gasoilkinetics(ne,f,ni,g,t,x(p_x),xdot(p_x),
+ ix(pix-pix0+1),pix,isel(p_s),is(pis-pis0+1),pis,
+ ipar(p_i),ii(pii-pii0+1),pii,rpar(p_r),ir(pir-pir0+1),pir,
+ iu1(iu1(l_u)),iu2(iu1(l_u)))
return
end ! subroutine m_gasoil
```

Using OpenAD

- Derivative code (forward) generated using OpenAD for these 16 examples

Gas and Oil' (16 KB), Bubble Point Calculation' (17 KB), Ethyl Acetate Reactor' (85 KB), Ethyl Acetate Reactor' (Variation 1) (85 KB), Lorenz Equations' (13 KB), Orthogonal Collocation Example' (19 KB), Phase Shift Oscillator (23 KB), Series Reactions' (12 KB), Dynamic Estimation Example' (7.5 KB), Simple Parameter Estimation Example' (8.4 KB), EGF Activation Model' (140 KB), A Continuous Distillation Column (416 KB)', A Batch Rectifier' (255 KB), A Simple Reactor Simulation' (52 KB), Another Reactor Simulation' (58 KB), A Binary Distillation Example' (244 KB)

- Several of them compiled and derivatives compared.
- Derivative codes via reverse mode generated.

Incorporating OpenAD with dSOA

- Second order derivatives: forward over reverse
- Generated code in FORTRAN 90
rest of the code in FORTRAN 77
- Testing with existing dSOA-(TN) examples
- Stand-alone vs. integrated