[Adjoint | Derivative] Code Design Patterns

Uwe Naumann
LuFG Informatik 12: Software and Tools for Computational Engineering (STCE), RWTH Aachen, Germany
in cooperation with The Numerical Algorithms Group (NAG) Ltd.
Motivation
Complexity ⇒ Pain

e.g., STCE/NAG @ AD2016

▶ Talks
  ▶ du Toit: AD in Finance
  ▶ Mosenkis: Elimination Techniques
  ▶ Towara: Adjoint MPI

▶ Posters
  ▶ Hüser: DAEO Sensitivities
  ▶ Lotz: Call Tree Reversal
  ▶ Sen: Adjoint OpenFOAM
  ▶ Leppkes: Adjoint NAG Library
Motivation
Adjoint-Driven Simulation Software Engineering

Algorithmic adjoint of \( F : \mathbb{R}^n \to \mathbb{R}^m : y \equiv z^q = F(x) \) evaluates

\[
\begin{align*}
\bar{X} \in \mathbb{R}^{n \times l} &= \nabla F^T, \\
\bar{Y} \in \mathbb{R}^{m \times l} &= \nabla F^1 T (\ldots \nabla F^k T (z^{k-1}) \cdot (\nabla F^{k+1} T \ldots \cdot (\nabla F^q T \cdot \bar{Y}) \ldots ) \ldots )
\end{align*}
\]

for elementals \( z_i = F_i (z_i^{i-1}) \) with Jacobians \( \nabla F^i (z_i^{i-1}) \) and \( x \equiv z^0 \).

The art of differentiating computer programs amounts to provision of correct, efficient, scalable, extensible, (insert your own desirable property) Adjoint Code Modules \( \bar{Z}_i^{i-1} = \nabla F^i T \cdot \bar{Z}_i \) for all elementals.

Design patterns formalize and document solutions for wide-spread relevant structural, algorithmic, and implementation problems in object-oriented software engineering; see “Gang of Four”.

\[1\] E. Gamma, R. Helm, R. Johnson, J. Vlissides: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1994. (23 classics)
Approach

Adjoint Code Design Patterns

E.g, dco/c++ adjoint (CLIENT) integrates hand-written adjoint (MyACModule)

Aim: Move as much logic (code) into ACDesignPatterns as possible / reasonable, e.g, ...
We consider an Euler-Maruyama scheme performing $\mu$ explicit Euler integration steps along $\nu$ independent Monte Carlo sample paths.\footnote{U. Naumann and J. du Toit: Adjoint Algorithmic Differentiation Tool Support for Typical Numerical Patterns in Computational Finance. To appear in Journal of Computational Finance.}

A basic implementation of this method evaluates a function of the following structure:

$$y = F(x) = G \left( \begin{array}{c} P_{\mu-1}^0 (P_{\mu-2}^0 (\ldots P_0^0 (S_0^0(x)) \ldots)) \\ \vdots \\ P_{\nu-1}^{\nu-1} (P_{\mu-2}^{\nu-1} (\ldots P_0^{\nu-1} (S_{\nu-1}^{\nu-1}(x)) \ldots)) \end{array} \right),$$

where the arguments $x$ (strike price, time to maturity, interest rate(s), price of underlying and market observed volatilities) are scattered over the individual paths by the $S_i^i$, $i = 0, \ldots, \nu - 1$, each path performs integration steps $P_j^i$, $j = 0, \ldots, \mu - 1$, and the results are gathered by $G$ to yield $y$ (option price).
Case Study
Solution based on ACDesignPatterns Library
Conclusion

Wouldn’t it be nice if ... 
 ▶ ... there was an ACDesignPatterns library covering common numerical and structural patterns in simulation code?
 ▶ ... potential clients (e.g., AD tools) would provide corresponding API?
 ▶ ... both developers and users of algorithmic adjoint patterns could benefit?

Admittedly, ... 
 ▶ ... (further) successful case studies required to convince most of you (dilemma: size matters)
 ▶ ... application to existing code can be nontrivial
 ▶ ... establishing (quasi-)standards likely to remain painful

Still, ... 
 ▶ ... dco/c++ support work in progress
 ▶ ... development ongoing, so stay tuned ... :-}
I thank Klaus Leppkes, Johannes Lotz, Jan Riehme, Viktor Mosenkis, Jonathan Hüser, and other members of STCE for challenging the general idea during various lively discussions.

\[^{3}\text{No questions by Klaus permitted today!}\]
Related Recent Publications


