Piecewise Linear AD via Source Transformation

Torsten Falko Bosse & Sri Hari Krishna Narayanan & Laurent Hascoët
Argonne National Laboratory, MCS Division &
9700 S. Cass Avenue
Argonne, IL 60439, US

INRIA Sophia-Antipolis
2004 Route des Lucioles, BP 93
06902 Valbonne, France

Assumption
Consider a piecewise smooth \( F : \mathbb{R}^n \to \mathbb{R}^m \), \( y = F(x) \), given by a finite evaluation procedure:

\[
\begin{align*}
\frac{\partial y}{\partial z}(x_{1,i}, x_{2,i}, \ldots, x_{n,i}) & \quad \text{for } i = 1, \ldots, p \quad \text{Initialization} \\
\frac{\partial y}{\partial z}(x_{1,i} + \Delta x, x_{2,i}, \ldots, x_{n,i}) & \quad \text{for } i = 1, \ldots, q \quad \text{Evaluation}
\end{align*}
\]

Here, \( x \in \mathbb{X} \subset \mathbb{R}^n \) is a vector of \( n \) input variables, \( y = g(x) \in \mathbb{R}^m \) are the corresponding \( m \) output variables, and \( g \) denotes some smooth intermediate assignments from a library \( \Phi \in \{ \ldots, \varphi(x, y, z), \ldots \} \) with partial derivatives \( \frac{\partial \varphi}{\partial x} \) and the absolute value function \( \|z\| = |z| \) for \( z \in \mathbb{R}^n \).

Piecwise Linear AD

The standard propagation rules for the standard forward mode of AD

\[
\begin{align*}
\Delta v_i &= \Delta v_j \pm \Delta g_i & \text{for } & v_i = v_j \mp v_k \\
\Delta v_i &= \Delta v_j - \Delta g_i & \text{for } & v_i = v_j \mp v_k \\
\Delta v_i &= \Delta v_j + \Delta g_i & \text{for } & v_i = v_j \mp v_k
\end{align*}
\]

can be extended to derive a piecewise linearization

\[
\Delta v_i = \Delta v_j \pm \Delta g_i & \quad \text{for } v_i = v_j \mp v_k \\
\Delta v_i &= \Delta v_j - \Delta g_i & \text{for } v_i = v_j \mp v_k \\
\Delta v_i &= \Delta v_j + \Delta g_i & \text{for } v_i = v_j \mp v_k
\]

of a piecewise smooth function \( F \) at a fixed base-point \( x \in \mathbb{R}^n \) and varying increment \( \Delta x \in \mathbb{R}^n \). Therefore, only the propagation rule for the absolute value needs to be adapted

\[
\Delta \|z\| = \Delta (\|z\|) - \|z\| \Delta (\|z\|) = \Delta (\|z\|) - \|z\| \Delta (\|z\|)
\]

Rules for the minimum and maximum can be easily derived from the absolute value.

Abs-normal form

When faced with a circuit with problematic components, an electrical engineer might want to temporarily isolate those components. Here, we will isolate each run-time call to abs by turning its input \( v \) into a new program’s output, and its output \( |v| \) into a new program’s input. This introduces an additional vector of switching variables \( \Delta v = \Delta (\|v\|) = \Delta (v, \|v\|) = \Delta (v, \|v\|) \in \mathbb{R}^n \), with \( n \) the number of run-time calls to \( \text{abs} \). This way, the calls to \( \text{abs} \) are pulled outside of the code to be differentiated by AD. The piecewise linearization \( \Delta F \) of \( F \) is now algebraically represented by a so-called “abs-normal form” (ANF)

\[
\begin{bmatrix}
\Delta 1 \\
\Delta 2 \\
\Delta 3 \\
\Delta 4
\end{bmatrix}
\]

where the matrix that expresses the linear dependencies is split into four blocks:

- \( J = \frac{\partial g}{\partial v} \in \mathbb{R}^{m \times n} \) the influence of the input \( x \) on the result \( y \) that don’t go through any abs call.
- \( Y = \frac{\partial g}{\partial \text{abs}} \in \mathbb{R}^{m \times n} \) the influence of the output of each abs call on the result \( y \), not going through any abs call.
- \( Z = \frac{\partial g}{\partial \text{abs}} \in \mathbb{R}^{m \times n} \) the influence of the input \( x \) on the inputs of each abs call, not going through any abs call.
- \( L = \frac{\partial g}{\partial \text{abs}} \in \mathbb{R}^{m \times n} \) the influence of the output of each abs call to the input of the following abs calls, not going through any other abs call.

These matrices are derivatives of code that only involve smooth expressions. They might obviously depend on \( x \). If the components of \( x \) are ordered in the chronological order of the calls to \( \text{abs} \), \( L \) is lower triangular. When a suitable use of AD has produced \( J, Y, Z \), and \( L \), \( \Delta F \) can be easily computed by evaluating the ANF from the top row down, and feeding each \( \Delta (\|z\|) \) as soon as it is computed.

Piecewise Linear AD and Source Transformation

To use the piecewise linear differentiation drivers in OpenAD or Tapenade, only the calls of \( \text{abs} \), \( \text{min} \), \( \text{max} \), and \( \text{sign} \) functions need to be differentiated in a special way. This is done classically through stub methods, whose AD-derivatives are automatically replaced by ad hoc derivative methods, described below.

In OpenAD for instance, we have the three stub methods

\[
\begin{align*}
gabs(u, a) & = \max(u, a), \\
gmax(z, v) & = \max(z, v), \\
gmin(z, v) & = \min(z, v)
\end{align*}
\]

The standard runtime library method for the tangent of e.g. \( gabs \), that preserves the original result of \( F \) but doesn’t compute the ANF yet, would be

\[
\begin{align*}
gabs(z, a) & = \max(z, a) - 0.5 \times (z - a)
\end{align*}
\]

Computation of the ANF using OpenAD

The ANF is computed by vector-tangent AD, that propagates directional derivatives in several directions simultaneously. Propagation follows rules (1). The number of directions, although initially equal to \( n \), must be able to grow up to \( n + 1 \) as the \( n \) calls to \( \text{abs} \) are encountered. With the active type approach of OpenAD, the derivatives must be stored in an array \( d \), of size large enough to accommodate \( n \times n \). We also define two global arrays \( dz \) and \( du \) that each hold, for each occurring absolute value, one vector of length \( n \).

Following our approach, the new runtime library method for the tangent of \( gabs \) that has taken away computation of the derivative \( u\text{abs} \), as it is pulled "outside" of the code to differentiate. Instead, the tangent derivatives of the input \( x \) are stored into \( dz \) as an additional output and the tangent derivative of the absolute value \( |z| \) are retrieved from the additional input \( dx \).

At the end of the tangent run, \( y \) holds \( J \) and \( Y \) side by side, and \( z \) holds \( Z \) and \( L \) side by side, as shown in Fig. 2.

Figure 1: Representation of a simple piecewise smooth function represented by its computational graph and a partitioning that is induced by absolute value calls in the evaluation procedure.

Figure 2: Storage layout for output/input of the tangent-differentiated code, showing the components of the ANF.

Standard Example

Consider the piecewise smooth function \( F : \mathbb{R}^2 \to \mathbb{R}, \, F(x, y) = \max(0, x - y - \max(x, y)) \) depicted in Fig. 2. The ANF of its piecewise linearization \( \Delta F = \Delta F(x, y) \) can be computed by evaluating the partial derivatives of the respective parts within its computational graph as indicated in Fig. 2.

The parts for the matrices \( Z, J, L \) and \( Y \) are colored in green, blue, purple, and yellow, respectively.

Figure 3: Visualization of the example and its piecewise linear model evaluated at the base point \( x = (0.1, 0.5) \).

Conclusions

The propagation rule for the absolute value, the maximum, and minimum function of OpenAD has been modified. This allows the evaluation of piecewise linearizations for piecewise smooth functions via source transformation. Furthermore, OpenAD is now able to compute the Abs-normal form representation with minimal additional user effort.

References


Acknowledgements

This material is based upon work supported by the U.S. Department of Energy, Office of Science, under contract number DE-AC02-06CH11357.