Différentiation Automatique avec TAPENADE

Laurent Hascoët

Projet TROPICS, INRIA Sophia-Antipolis

January 16th, 2003

A.D. in brief

Given mathematical equations and a program (discretization), **how** can we obtain derivatives ?

- Write new equations, then discretize boldly
- Start from the given program, and
  * compute divided differences
  * (A.D.) transform the program to compute analytical derivatives
    - either by overloading
    - or by creating new instructions

Using the Chain Rule

Consider a program $P : \{I_1; I_2; \ldots I_p;\}$ implementing $f : \mathbb{R}^m \to \mathbb{R}^n$
We identify programs with composition of functions, one per instruction:
$f = f_p \circ f_{p-1} \circ \cdots \circ f_1$ (f_k implemented by I_k). The chain rule says:

$$
\begin{aligned}
f'(x) \quad = \quad & (f'_p \circ f_{p-1} \circ f_{p-2} \circ \cdots \circ f_1(x)) \\
& .\,(f'_{p-1} \circ f_{p-2} \circ \cdots \circ f_1(x)) \\
& .\,\ldots \\
& .\,(f'_1(x)) \\
= \quad & f'_p(x_{p-1}).f'_{p-1}(x_{p-2}).\ldots.f'_1(x_0)
\end{aligned}
$$

Example

$$
\begin{aligned}
& \dots \\
v_2 &= 2 * v_1 + 5 \\
v_4 &= v_2 + p_1 * v_3 / v_2 \\
& \dots
\end{aligned}
$$

The corresponding (fragment of) Jacobian is:

$$
Jac = \dots \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ 0 & 1 - \frac{p_1 * v_3}{v_2^2} & \frac{p_1}{v_2} & 0 \end{pmatrix} \begin{pmatrix} 1 & & & \\ 2 & 0 & & \\ & & 1 & \\ & & & 1 \end{pmatrix} \dots
$$

Tangent mode

We might need...

•**sensitivity:** $\dot{y} = f'.\dot{x}$ leads to the tangent mode

Sensitivity is computed by simple augmentation of the program:

$$\begin{array}{rcl} & & \ldots \\ \dot{v}_2 & = & 2 * \dot{v}_1 \\ v_2 & = & 2 * v_1 \ + \ 5 \\ \dot{v}_4 & = & \dot{v}_2 * (1 - p_1 * v_3/v_2^2) \ + \ \dot{v}_3 * p_1/v_2 \\ v_4 & = & v_2 \ + \ p_1 * v_3/v_2 \\ & & \ldots \end{array}$$

Reverse mode

We might need...

•**gradient:** $\overline{x}^* = \overline{y}^*.f'$ leads to the reverse mode

$$\begin{aligned}\overline{x} &= f'^*(x).\overline{y} \\ &= f_1'^*(x_0).f_2'^*(x_1)\dots f_{p-1}'^*(x_{p-2}).f_p'^*(x_{p-1}).\overline{y}\end{aligned}$$

$\overline{x}$ is evaluated in the reverse order of the original program's result $y = x_p$

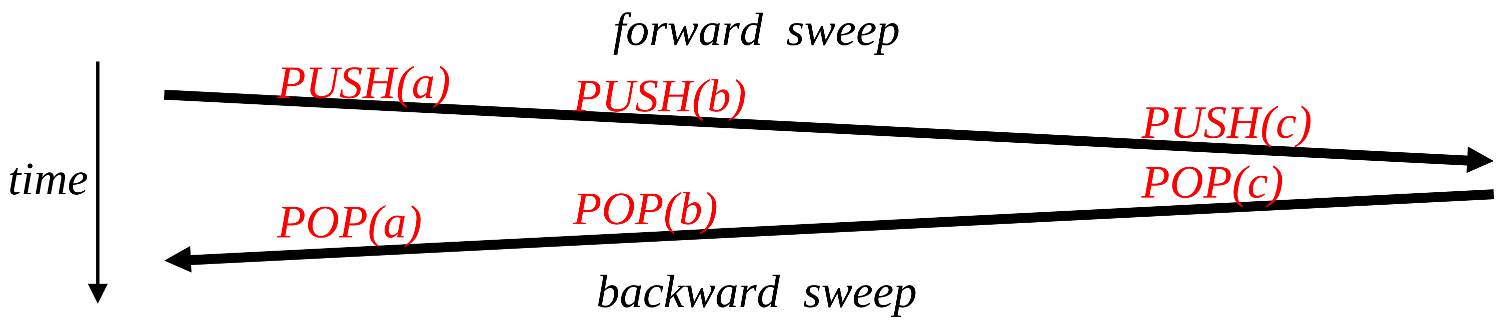
Reverse mode on the example

$$
\begin{array}{rcl}
 & & \dots \\
v_2 & = & 2 * v_1 \; + \; 5 \\
v_4 & = & v_2 \; + \; p_1 * v_3 / v_2 \\
 & & \dots
\end{array}
$$

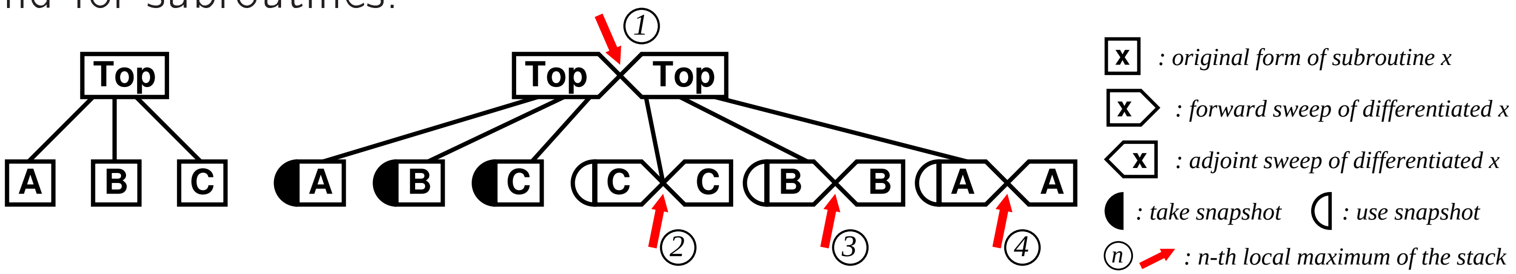
$$
\begin{array}{rcl}
 & & \dots \\
\bar{v}_2 & = & \bar{v}_2 \; + \; \bar{v}_4 * (1 - p_1 * v_3 / v_2^2) \\
\bar{v}_3 & = & \bar{v}_3 \; + \; \bar{v}_4 * p_1 / v_2 \\
\bar{v}_4 & = & 0 \\
\bar{v}_1 & = & \bar{v}_1 \; + \; 2 * \bar{v}_2 \\
\bar{v}_2 & = & 0
\end{array}
$$

$\Rightarrow$ Notice that v_2 and v_3 may need to be saved !

Model for the reverse mode



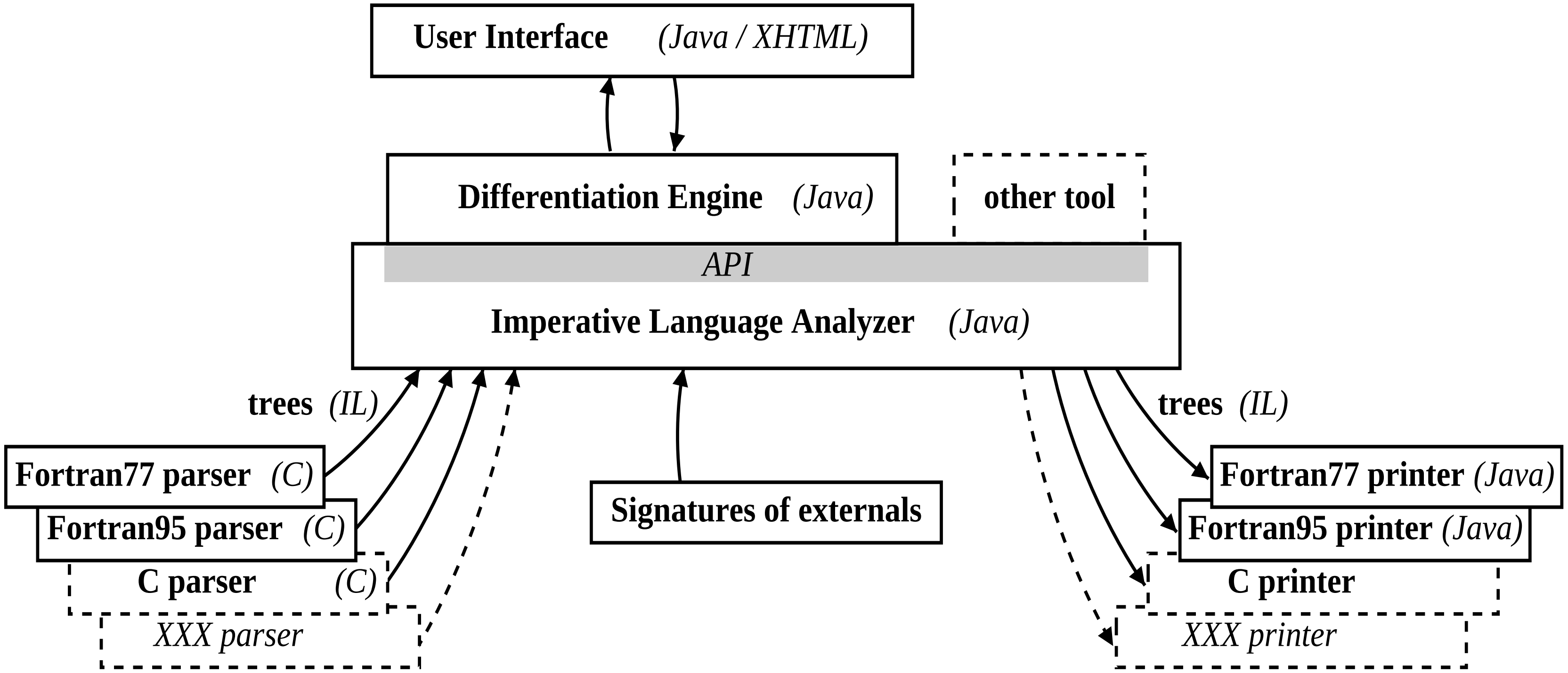
And for subroutines:



The bottleneck is memory size !!

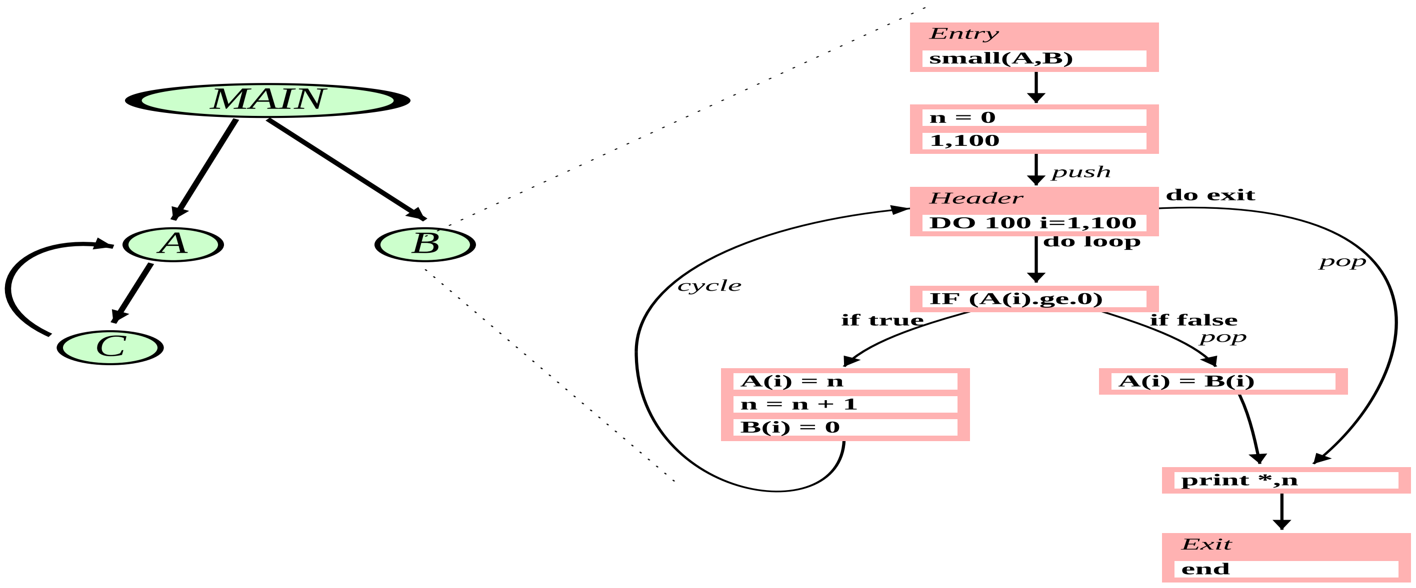
TAPENADE: An A.D. Tool on the Web

Tangent and Adjoint Automatic Differentiation Engine



TAPENADE: Internal Representation

Call-Graph of Flow-Graphs



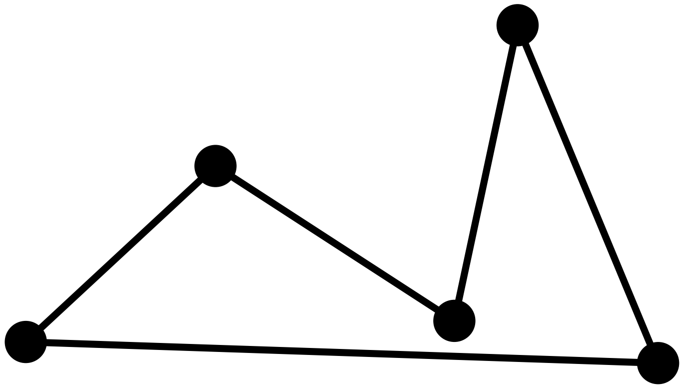
- Nested Symbol Tables for scoping
- Built-in mechanism for translation of variable-vased info across call boundaries

TAPENADE: Focus on the Reverse Mode

- Bi-directional Interprocedural Activity analysis.
- Interprocedural TBR analysis.
- Interprocedural In-Out analysis for smaller Snapshots.
- Control Reversal on the Flow Graph.
- Global Tape (reduces snapshots).
- Checkpointing on calls, with option to inhibit.
- Local optimizations on adjoint Basic Blocks.

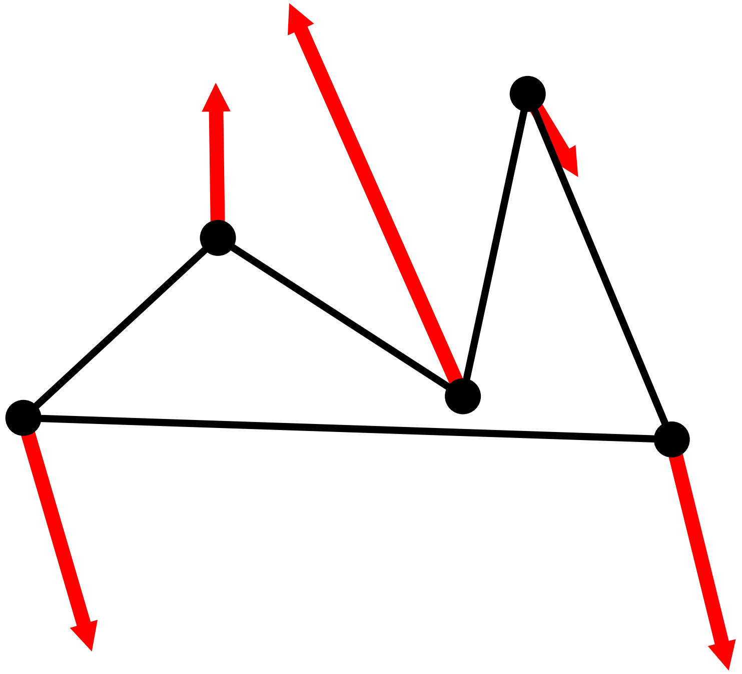
A toy Optimization Problem

Move the vertices of a polygon to minimize the L^2/S ratio.



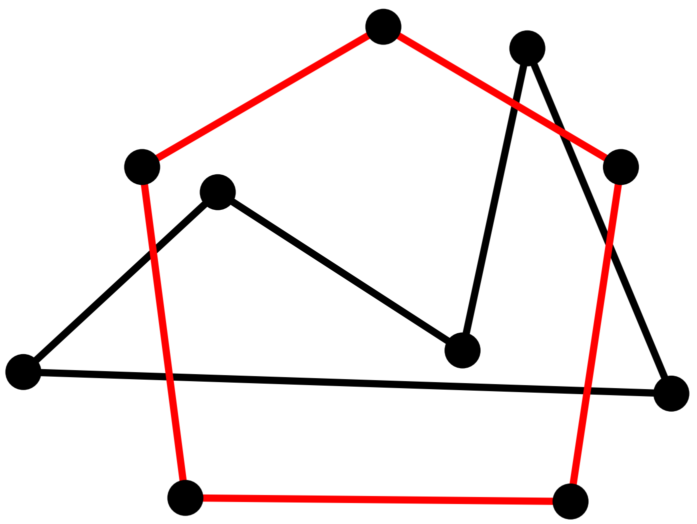
Gradient by AD

The Gradient is obtained in a single run of the function.



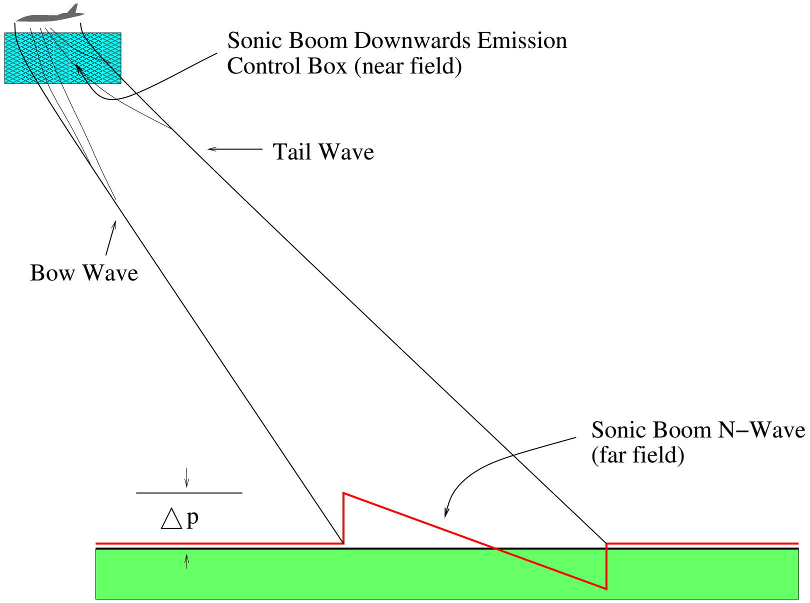
After 30 iterations...

The polygon stabilized to:



The Sonic Boom problem

Modify the control points that define the shape of a supersonic aircraft, to reduce the sonic boom felt on the ground.



A Hybrid Approach

Use the adjoint equations, write a new solver, but parts will be done with AD.

$$L(W,\gamma,\Pi) \;=\; J(W,\gamma) + \langle \Psi(\gamma,W),\Pi\rangle, \tag{1}$$

leads to the system:

$$\left\{\begin{array}{rcll} \Psi(\gamma,W) & = & 0 & \textit{(state system)} \\[2ex] \dfrac{\partial J}{\partial W}(\gamma,W) - (\dfrac{\partial \Psi}{\partial W}(\gamma,W))^t \,.\, \Pi & = & 0 & \textit{(adjoint system)} \\[2ex] j'(\gamma) = \dfrac{\partial J}{\partial \gamma}(\gamma,W) - (\dfrac{\partial \Psi}{\partial \gamma}(\gamma,W))^t \,.\, \Pi & = & 0 & \textit{(optimality condition)} \end{array}\right. \tag{2}$$

Validation of AD derivatives

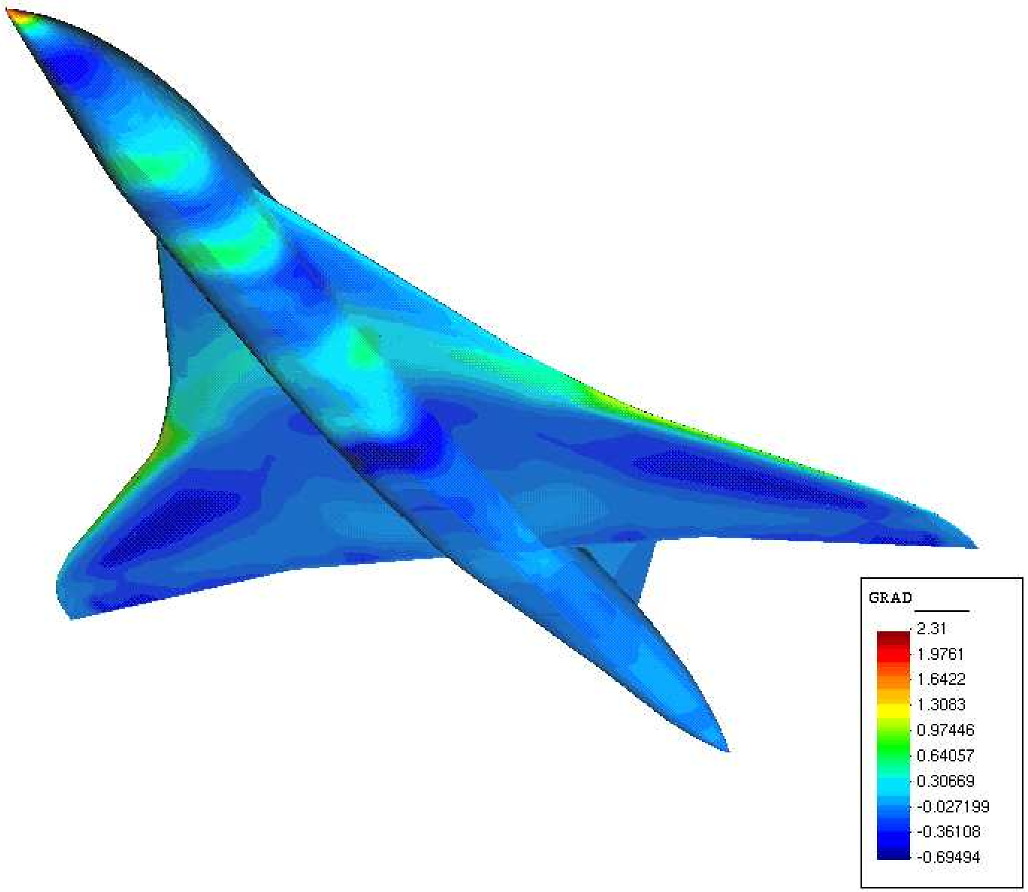
Tangent: $\dot{y} = f'(x).\dot{x}$ validated wrt Divided Differences

Reverse: $\overline{x} = f'(x)^*.\overline{y}$ validated wrt Tangent by setting $\overline{y}$ to $\dot{y}$ and remarking that:

$$< \overline{x} \ , \ \dot{x} > = < f'(x)^*.\dot{y} \ , \ \dot{x} > = < \dot{y} \ , \ f'(x).\dot{x} > = < \dot{y} \ , \ \dot{y} >$$

Squared Divided Differences	$< \dot{y} \ , \ \dot{y} >$	$< \overline{x} \ , \ \dot{x} >$
3301.12804	3301.12817	3301.12817

Gradient on the skin of the plane



Optimization Results