Mechanical Generation of Algorithms for Automatic Differentiation

Diego Fabregat-Traver

Advisor: Prof. Paolo Bientinesi

Automatic Generation of Algorithms

Our methodology for automatically generating algorithms is based on formal methods. Given a formal description of a target operation, we perform a series of symbolic steps to obtain a set of predicates called loop-invariants. Each loop-invariant then leads to a corresponding algorithm. The process is completely mechanical and can be automatically performed by computer algebra systems.

Automatic Differentiation

Automatic Differentiation is a method to numerically evaluate the derivative of a function specified by a computer program. Through a process called Source Code Transformation, the source code implementing a function is automatically augmented to compute both the function and its derivative with respect to one or more variables. This technique can be applied in perturbation analysis, optimization, ...

Input

The formal description of the target operation is given by two predicates, the **Precondition** and the **Postcondition**. In this case:

$$f(\alpha, A, B, X) = 0 \equiv \begin{cases} f_{\text{Pre}}: & \{Input(\alpha, A, B) \wedge \\ & LowTri(A) \wedge \\ & Output(X)\} \\ f_{\text{Post}}: & \{AX = \alpha B\} \end{cases}$$

$$f(\alpha, A, B, X) = 0 \quad \equiv \quad AX = \alpha B$$

$$\frac{df}{dv} = ?$$

$$A'X + AX' = \alpha' B + \alpha B' \quad \dots \quad A'X + AX' = \alpha' B \quad \dots \quad AX' = 0$$

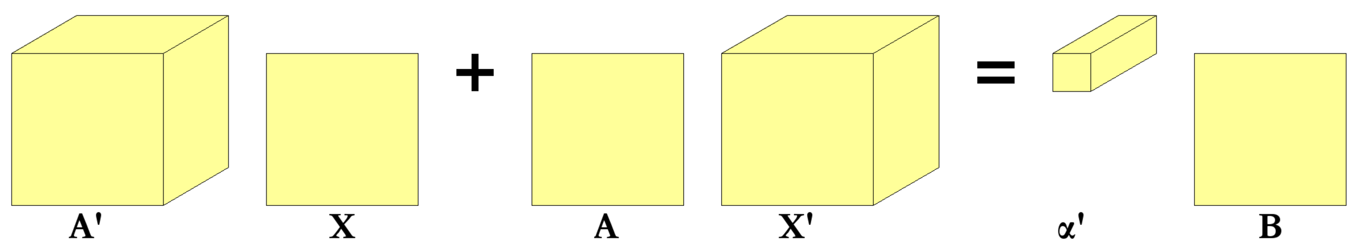
Kernels

According to the functional dependency of the operands with respect to v, **multiple computational kernels** are required.

Precondition and Postcondition

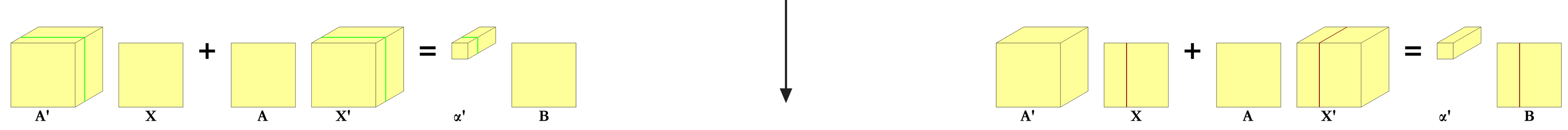
Each derivative kernel has its own pair of Precondition and Postcondition predicates. In the case of $A'X + AX' = \alpha' B$:

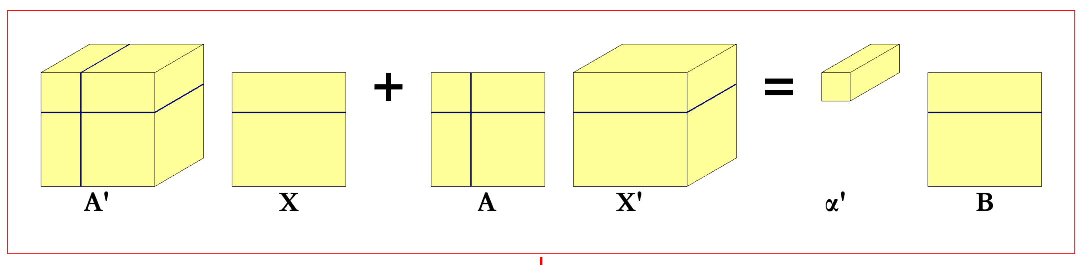
$$\begin{array}{ll} \textit{Precondition}: & \{Input(\alpha', A, A', B, X) \wedge \\ & LowTri(A, A') \wedge \\ & Output(X')\} \\ \textit{Postcondition}: & \{A'X + AX' = \alpha' B\} \end{array}$$



3D arrays

The differentiation with respect to multiple variables (array v), results in **three dimensional arrays**.





Partitionings

All the possible combinations of **partitionings** for the operands are automatically identified.

Partitioned Postcondition

The **operands** in the postcondition are **replaced by** their **partitioned counterparts**.

$$\left(\begin{array}{c|c} A'_{TL} & 0 \\ \hline A'_{BL} & A'_{BR} \end{array}\right) \left(\frac{X_T}{X_B}\right) + \left(\begin{array}{c|c} A_{TL} & 0 \\ \hline A_{BL} & A_{BR} \end{array}\right) \left(\frac{X'_T}{X'_B}\right) = \alpha' \left(\frac{B_T}{B_B}\right)$$

PME derivation

Several steps of **algebraic manipulation and pattern matching** are automatically performed until the PME (Partitioned Matrix Expression) is obtained.

PME

The **PME** is at the core of the methodology. It is a **recursive definition of the operation** representing how the different parts of the output are computed in terms of parts of the input. It also shows which **operations** are **to be performed** in each quadrant **and** the **dependencies** between said operations.

$$\left(\frac{X'_T = A_{TL}^{-1}(\alpha' B_T - A'_{TL} X_T)}{X'_B = A_{BR}^{-1}(\alpha' B_B - A'_{BL} X_T - A'_{BR} X_B - A_{BL} X'_T)}\right)$$

$$\left(\frac{X'_T = A_{TL}^{-1}(\alpha' B_T - A'_{TL} X_T)}{X'_B = \alpha' B_B - A'_{BL} X_T - A'_{BR} X_B - A_{BL} X'_T}\right) \quad \dots \quad \left(\frac{X'_T = A_{TL}^{-1}(\alpha' B_T - A'_{TL} X_T)}{X'_B = \alpha' B_B - A'_{BR} X_B - A_{BL} X'_T}\right) \quad \dots \quad \left(\frac{X'_T = A_{TL}^{-1}(\alpha' B_T - A'_{TL} X_T)}{X'_B = B_B}\right)$$

Loop-Invariants

From the PME a family of loop-invariants are automatically derived. A loop invariant is a predicate that is true at the beginning and the end of each iteration of a loop. A loop-invariant constitutes the skeleton of a proof of correctness around which the final algorithm is automatically built.

Loop-Invariants to Algorithms

There is a **one to one correspondence** between loop-invariants and algorithms.

Partition $A' \rightarrow \left(\begin{array}{c|c} A'_{TL} & 0 \\ \hline A'_{BL} & A'_{BR} \end{array}\right), X \rightarrow \left(\frac{X_T}{X_B}\right), \dots$

where A'_{TL} is $0 \times 0 \times k$, X_T is $0 \times n$, ...

While $n(X_T) < n(X')$ **do**

Repartition $\left(\begin{array}{c|c} A'_{TL} & 0 \\ \hline A'_{BL} & A'_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A'_{00} & 0 & 0 \\ \hline A'_{10} & A'_{11} & 0 \\ \hline A'_{20} & A'_{21} & A'_{22} \end{array}\right), \left(\frac{X_T}{X_B}\right) \rightarrow \left(\begin{array}{c} X_0 \\ \hline X_1 \\ \hline X_2 \end{array}\right), \dots$

where A'_{11} is $1 \times 1 \times k$, X_1 is $1 \times n$, ...

$X'_1 := \alpha' B_1$	(GER)
$X'_1 := X'_1 - A'_{10} X_0$	(GEMM)
$X'_1 := X'_1 - A'_{11} X_1$	(GER)
$X'_1 := X'_1 - A_{10} X'_0$	(multiple GEMVs)
$X'_1 := A_{11}^{-1} X'_1$	(SCAL)

Continue with $\left(\begin{array}{c|c} A'_{TL} & 0 \\ \hline A'_{BL} & A'_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A'_{00} & 0 & 0 \\ \hline A'_{10} & A'_{11} & 0 \\ \hline A'_{20} & A'_{21} & A'_{22} \end{array}\right), \left(\frac{X_T}{X_B}\right) \leftarrow \left(\begin{array}{c} X_0 \\ \hline X_1 \\ \hline X_2 \end{array}\right), \dots$

endwhile

Algorithms

One of the **algorithms for computing the operation** $A'X + AX' = \alpha' B$. **Every step in the methodology** for building the algorithm **is fully automated**.

Funding from DFG through grant GSC 111 is gratefully acknowledged

Deutsche Forschungsgemeinschaft

DFG

AACHEN INSTITUTE FOR ADVANCED STUDY IN COMPUTATIONAL ENGINEERING SCIENCE

RWTH AACHEN UNIVERSITY

Graduate School AICES · RWTH Aachen University · Templergraben 55 · 52056 Aachen, Germany · Tel.: +49(0)241/80-28498 · www.aices.rwth-aachen.de