Generalized Jacobian Chain Products

Uwe Naumann

Informatik 12:
Software and Tools for Computational Engineering (STCE)

RWTH Aachen University

Contents

Outline

Software and Tools for Computational Engineering
RWTH AACHEN UNIVERSITY

Objective and Learning Outcomes

Prerequisites
- Chain Rule
- Tangents and Adjoints
- DAG
- Vector Tangents and DAG × Matrix Products
- Vector Adjoints and Matrix × DAG Products

Generalized Jacobian Chain Product

Generalized Dense Jacobian Chain Product Bracketing
- Dynamic Programming
- Implementation

Summary and Next Steps

Objective

- Introduction to Generalized Jacobian Chain Products as a milestone towards cost-optimal (algorithmic) differentiation.

Learning Outcomes

- You will understand
  - GENERALIZED DENSE JACOBIAN CHAIN PRODUCT problem in unlimited memory
  - dynamic programming algorithm for its solution
- You will be able to
  - download and build the GDJCPB software.
  - run your own experiments.

Outline

Objective and Learning Outcomes

Prerequisites
- Chain Rule
- Tangents and Adjoints
- DAG
- Vector Tangents and DAG × Matrix Products
- Vector Adjoints and Matrix × DAG Products

Generalized Jacobian Chain Product

Generalized Dense Jacobian Chain Product Bracketing
- Dynamic Programming
- Implementation

Summary and Next Steps

Let the primal function

$$y = F(x) : \boldsymbol{R}^n \rightarrow \boldsymbol{R}^m$$

be continuously differentiable over the domain of interest and let

$$F = F_q \circ F_{q-1} \circ \ldots \circ F_2 \circ F_1$$

be such that $z_i = F_i(z_{i-1}) : \boldsymbol{R}^{n_i} \rightarrow \boldsymbol{R}^{m_i}$ for $i = 1, \ldots, q$ and $z_0 = x$, $y = z_q$.

According to the chain rule of differential calculus the Jacobian $F' = F'(x)$ of F is equal to the result of the Jacobian chain product

$$F' \equiv \frac{dF}{dx} = F'_q \cdot F'_{q-1} \cdot \ldots \cdot F'_1 \in \boldsymbol{R}^{m \times n} \ . \tag{1}$$

We denote the computational cost in term of *fused multiply-add* (`fma`) operations of evaluating a subchain $F'_j \cdot \ldots \cdot F'_i$, $j > i$, as $\texttt{fma}_{j,i}$.

Algorithmic differentiation offers two fundamental modes for preaccumulation of the local Jacobians $F_i' = F_i'(z_{i-1}) \in \boldsymbol{R}^{m_i \times n_i}$ prior to the evaluation of the above Jacobian chain product:

- Tangent mode

$$\dot{z}_i = F_i' \cdot \dot{z}_{i-1} \in \boldsymbol{R}^{m_i} \; .$$

  Accumulation of a dense Jacobian requires evaluation of n_i tangents in the Cartesian basis directions in $\boldsymbol{R}^{n_i}$. The computational cost of evaluating F_i' in tangent mode is denoted as $\texttt{fma}_{i,i} = \dot{\texttt{fma}}_i$.

- Adjoint mode

$$\bar{z}_{i-1} = \bar{z}_i \cdot F_i' \in \boldsymbol{R}^{1 \times n_i}$$

  and hence dense Jacobians by m_i evaluations with $\bar{z}_i$ ranging over the Cartesian basis directions in $\boldsymbol{R}^{m_i}$. The computational cost of evaluating F_i' in adjoint mode is denoted as $\texttt{fma}_{i,i} = \bar{\texttt{fma}}_i$.

The Jacobian Chain Product problem asks for an `fma`-optimal evaluation of the right-hand side of Equation (1).

As a variant of Sparse Matrix Chain Product it is known to be NP-complete.

See also modules on [Sparse] Matrix Chain Products and on Elimination Methods on DAGs.

Prerequisites

Jacobian Chain Product Bracketing

The Jacobian Chain Product Bracketing problem asks for a bracketing of the right-hand side of Equation (1) which minimizes the number of fma operations.

As a variant of [Sparse] Matrix Chain Product Bracketing it can be solved by dynamic programming taking into account minimal preaccumulation cost of the individual factors, i.e,

$$\mathtt{fma}_{j,i} = \begin{cases} \min(\dot{\mathtt{fma}}_i, \bar{\mathtt{fma}}_i) & j = i \\ \min_{i \le k < j} \left(\mathtt{fma}_{j,k+1} + \mathtt{fma}_{k,i} + \mathtt{fma}_{j,k,i}\right) & j > i\,. \end{cases}$$

The $F_i = F_i(z_{i-1})$ induce labeled directed acyclic graphs (DAGs)

$$G_i = G_i(z_{i-1}) = (V_i, E_i)$$

for $i = 1, \ldots, q$. Vertices in $V_i = \{v_j^i : j = 1, \ldots, |V_i|\}$ represent the elemental arithmetic operations $\varphi_j^i \in \{+, \sin, \ldots\}$ executed by the implementation of F_i for given z_{i-1}. Edges in $(j,k) \in E_i \subseteq V_i \times V_i$ mark data dependencies between arguments and results of elemental operations. They are labeled with local partial derivatives

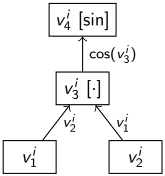
$$\frac{\partial \varphi_k^i}{\partial v_j^i}, \quad k : (j,k) \in E_i$$

of the elemental functions with respect to their arguments.

See also modules on Calculus.

DAGs, Tangents, Adjoints

Example



(a)

$$
\begin{aligned}
v_1^i &= z_1^i \\
v_2^i &= z_2^i \\
v_3^i &= v_1^i \cdot v_2^i \\
v_4^i &= \sin(v_3^i) \\
z_1^{i+1} &= v_4^i
\end{aligned}
$$

(b)

$$
\begin{aligned}
\dot{v}_1^i &= \dot{z}_1^i \\
\dot{v}_2^i &= \dot{z}_2^i \\
\dot{v}_3^i &= v_2^i \cdot \dot{v}_1^i + v_1^i \cdot \dot{v}_2^i \\
\dot{v}_4^i &= \cos(v_3^i) \cdot \dot{v}_3^i \\
\dot{z}_1^{i+1} &= \dot{v}_4^i
\end{aligned}
$$

(c)

$$
\begin{aligned}
\bar{v}_4^i &= \bar{z}_1^{i+1} \\
\bar{v}_3^i &= \bar{v}_4^i \cdot \cos(v_3^i) \\
\bar{v}_2^i &= \bar{v}_3^i \cdot v_1^i \\
\bar{v}_1^i &= \bar{v}_3^i \cdot v_2^i \\
\bar{z}_2^i &= \bar{v}_2^i \\
\bar{z}_1^i &= \bar{v}_1^i
\end{aligned}
$$

(d)

Labeled DAG (a); primal (b); tangent (c); adjoint (d)

See also modules on Algorithmic Differentiation.

For given $z_{i-1} \in \boldsymbol{R}^{n_i}$ and $\dot{Z}_{i-1} \in \boldsymbol{R}^{n_i \times \dot{n}_i}$ the Jacobian-free evaluation of

$$\dot{Z}_i = F_i'(z_{i-1}) \cdot \dot{Z}_{i-1} \in \boldsymbol{R}^{m_i \times \dot{n}_i}$$

in vector tangent mode is denoted as

$$\dot{Z}_i := \dot{F}_i(z_{i-1}) \cdot \dot{Z}_{i-1} \ . \qquad (2)$$

Preaccumulation of a dense F_i' requires $\dot{Z}_{i-1}$ to be equal to the identity $I_{n_i} \in \boldsymbol{R}^{n_i \times n_i}$. Equation (2) amounts to the simultaneous propagation of $\dot{n}_i$ tangents through G_i.

Tangent propagation induces a computational cost of $\dot{n}_i \cdot |E_i|$.

Explicit construction (and storage) of G_i is not required.

Equation (2) can be interpreted as the "product" of the DAG G_i with the matrix $\dot{Z}_{i-1}$.

For given $z_{i-1} \in \mathbb{R}^{n_i}$ and $\bar{Z}_i \in \mathbb{R}^{\bar{m}_i \times m_i}$ the Jacobian-free evaluation of

$$\bar{Z}_{i-1} = \bar{Z}_i \cdot F_i'(z_{i-1}) \in \mathbb{R}^{\bar{m}_i \times n_i}$$

in vector adjoint mode is denoted as

$$\bar{Z}_{i-1} := \bar{Z}_i \cdot \bar{F}_i(z_{i-1}) \,. \tag{3}$$

Preaccumulation of a dense F_i' requires $\bar{Z}_i$ to be equal to the identity $I_{m_i} \in \mathbb{R}^{m_i \times m_i}$.

Vector adjoint propagation induces an computational cost of $\bar{m}_i \cdot |E_i|$.

Explicit construction (and storage) of G_i is required (potentially infeasible memory requirement).

Equation (3) can be interpreted as the “product” of the matrix $\bar{Z}_i$ with the DAG G_i.

Outline

The GENERALIZED JACOBIAN CHAIN PRODUCT (GJCP) problem asks for an algorithm for computing F' with a minimum number of `fma` operations for given tangents and adjoints for all F_i in Equation (1).

As a variant of JACOBIAN CHAIN PRODUCT it is known to be NP-complete. See module on [Sparse] Matrix Chain Products.

Outline

We consider the GENERALIZED DENSE JACOBIAN CHAIN PRODUCT BRACKETING (GDJCPB) problem in unlimited memory (no memory constraints).

Let tangents $\dot{F}_i \cdot \dot{Z}_i$ and adjoints $\bar{Z}_{i+1} \cdot \bar{F}_i$ be given for all elemental functions F_i, $i = 1, \ldots, q$, in Equation (1) whose respective Jacobians are assumed to be dense.

For a given positive integer K is there a sequence of evaluations of the tangents and/or adjoints such that the number of `fma` operations required for the accumulation of the Jacobian F' is less than or equal to K?

An instance of GDJCPB of length two yields the following eight different bracketings:

- $F' = \dot{F}_2 \cdot F_1' = \dot{F}_2 \cdot (\dot{F}_1 \cdot I_{n_0})$ (homogeneous tangent)
- $F' = \dot{F}_2 \cdot F_1' = \dot{F}_2 \cdot (I_{n_1} \cdot \bar{F}_1)$
- $F' = F_2' \cdot \bar{F}_1 = (I_{n_2} \cdot \bar{F}_2) \cdot \bar{F}_1$ (homogeneous adjoint)
- $F' = F_2' \cdot \bar{F}_1 = (\dot{F}_2 \cdot I_{n_1}) \cdot \bar{F}_1$
- $F' = F_2' \cdot F_1' = (\dot{F}_2 \cdot I_{n_1}) \cdot (I_{n_1} \cdot \bar{F}_1)$ (homogeneous preaccumulation)
- $F' = F_2' \cdot F_1' = (I_{n_2} \cdot \bar{F}_2) \cdot (I_{n_1} \cdot \bar{F}_1)$ (homogeneous preaccumulation)
- $F' = F_2' \cdot F_1' = (I_{n_2} \cdot \bar{F}_2) \cdot (\dot{F}_1 \cdot I_{n_0})$ (homogeneous preaccumulation)
- $F' = F_2' \cdot F_1' = (\dot{F}_2 \cdot I_{n_1}) \cdot (\dot{F}_1 \cdot I_{n_0})$. (homogeneous preaccumulation)

$$
\mathtt{fma}_{j,i} = \begin{cases} |E_j| \cdot \min\{n_j, m_j\} & j = i \\ \min_{i \le k < j} \left\{ \min \begin{Bmatrix} \mathtt{fma}_{j,k+1} + \mathtt{fma}_{k,i} + m_j \cdot m_k \cdot n_i, \\ \mathtt{fma}_{j,k+1} + m_j \cdot \sum_{\nu=i}^{k} |E_\nu|, \\ \mathtt{fma}_{k,i} + n_i \cdot \sum_{\nu=k+1}^{j} |E_\nu| \end{Bmatrix} \right\} & j > i\,. \end{cases}
$$

See U.N.: Optimization of Generalized Jacobian Chain Products without Memory Constraints. arXiv:2003.05755 [math.NA], 2020 for proof.

Example

$$n_1 = 4,\ m_1 = n_2 = 2,\ m_2 = 32,\ |E_1| = |E_2| = 100$$

$$\texttt{fma}\left(\dot{F}_2 \cdot (\dot{F}_1 \cdot I_{n_1})\right) = 800, \quad \texttt{fma}\left(\dot{F}_2 \cdot (I_{m_1} \cdot \bar{F}_1)\right) = 600,$$

$$\texttt{fma}\left((I_{m_2} \cdot \bar{F}_2) \cdot \bar{F}_1\right) = 6400, \quad \texttt{fma}\left((\dot{F}_2 \cdot I_{n_2}) \cdot \bar{F}_1\right) = 3400,$$

$$\texttt{fma}\left((\dot{F}_2 \cdot I_{n_2}) \cdot (I_{m_1} \cdot \bar{F}_1)\right) = 656, \quad \texttt{fma}\left((I_{m_2} \cdot \bar{F}_2) \cdot (I_{m_1} \cdot \bar{F}_1)\right) = 3656,$$

$$\texttt{fma}\left((I_{m_2} \cdot \bar{F}_2) \cdot (\dot{F}_1 \cdot I_{n_1})\right) = 3856, \quad \texttt{fma}\left((\dot{F}_2 \cdot I_{n_2}) \cdot (\dot{F}_1 \cdot I_{n_1})\right) = 856.$$

Software and Tools for Computational Engineering RWTH AACHEN UNIVERSITY

Consider an instance of GDJCPB of length $q = 3$ with

1. $n_1 = 3,\ m_1 = 3,\ |E_1| = 29$
2. $n_2 = 3,\ m_2 = 1,\ |E_2| = 14$
3. $n_3 = 1,\ m_3 = 2,\ |E_3| = 7$

Optimal preaccumulation of the local Jacobians induces the following costs:

1. $F'_1 :\ \texttt{fma}_{1,1} = 87$
2. $F'_2 :\ \texttt{fma}_{2,2} = 14$
3. $F'_3 :\ \texttt{fma}_{3,3} = 7$

Two subproblems of length two need to be considered:

- $F'_{3,2} \equiv F'_3 \cdot F'_2$ comprising
  - $F'_3 \cdot F'_2$ at $7 + 14 + 3 \cdot 1 \cdot 2 = 27\texttt{fma}$
  - $F'_3 \cdot \bar{F}_2$ at $7 + 2 \cdot 14 = 35\texttt{fma}$
  - $\dot{F}_3 \cdot F'_2$ at $14 + 3 \cdot 7 = 35\texttt{fma}$

  and, hence, yielding $\texttt{fma}_{3,2} = 27$.
- $F'_{2,1} \equiv F'_2 \cdot F'_1$ comprising
  - $F'_2 \cdot F'_1$ at $87 + 14 + 3 \cdot 3 \cdot 1 = 110\texttt{fma}$
  - $F'_2 \cdot \bar{F}_1$ at $14 + 1 \cdot 29 = 43\texttt{fma}$
  - $\dot{F}_2 \cdot F'_1$ at $87 + 3 \cdot 14 = 129\texttt{fma}$

  and, hence, yielding $\texttt{fma}_{3,2} = 43$.

The last iteration compares

- $F'_{3,2} \cdot F'_1$ at $27 + 87 + 2 \cdot 3 \cdot 3 = 132\texttt{fma}$
- $F'_{3,2} \cdot \bar{F}_1$ at $27 + 2 \cdot 29 = 85\texttt{fma}$
- $\dot{F}_{3,2} \cdot F'_1$ at $87 + 3 \cdot (7 + 14) = 150\texttt{fma}$
- $F'_3 \cdot F'_{2,1}$ at $7 + 43 + 2 \cdot 1 \cdot 3 = 56\texttt{fma}$
- $F'_3 \cdot \bar{F}_{2,1}$ at $7 + 2 \cdot (14 + 29) = 93\texttt{fma}$
- $\dot{F}_3 \cdot F'_{2,1}$ at $43 + 3 \cdot 7 = 64\texttt{fma}$

yielding

$$\texttt{fma}_{3,1} = 56 \; .$$

Go to www.github.com/un110076/ADMission/GDJCPB for

- gdjcpb_generate.exe generates problem instances randomly for a given length len of the chain and upper bound max_m_n on the number of rows and columns of the individual factors.
- gdjcpb_solve.exe computes one solution to the given problem instance.

len	max_mn	Tangent	Adjoint	Preaccumulation	Optimum
10	10	3,708	5,562	2,618	1,344
50	50	1,283,868	1,355,194	1,687,575	71,668
100	100	3,677,565	44,866,293	40,880,996	1,471,636
250	250	585,023,794	1,496,126,424	1,196,618,622	9,600,070
500	500	21,306,718,862	19,518,742,454	1,027,696,225	149,147,898

Table: Test Results: Cost in fma

Outline

Summary

- GENERALIZED DENSE JACOBIAN CHAIN PRODUCT problem in unlimited memory
- dynamic programming algorithm for GDJCPB
- GDJCPB software on www.github.com

Outlook

- Sparsity and memory constraints should be taken into account.

Next Steps

- Work through examples in paper on arXiv.
- Validate using GDJCPB software.
- Continue the course to find out more ...