# Derivation of Augmented Arithmetic for Computing Gradient, Hessian and Jacobian through Forward Mode AD Using Dual Numbers 

P.Senthil Vadivu (Corresponding author)<br>Department of Mathematics<br>Sona College of Technology<br>Salem-636005, Tamil Nadu, India

Tel: 91-427-233-6383 E-mail: psvoct@gmail.com
S. Ponnusamy

Department of Mathematics
Sona College of Technology
Salem-636005, Tamil Nadu, India
Tel: 91-427-409-9783 E-mail: ponsam@yahoo.com


#### Abstract

This paper presents a new approach to Automatic Differentiation (AD) for a scalar valued and twice continuously differentiable function $f: \mathbb{R}^{n} \rightarrow \mathbb{R}$. A new arithmetic is obtained based on the chain rule and using augmented algebra of real numbers. The chain rule based differentiation arithmetic is used to find the Gradient and Hessian. Jacobian is evaluated using Gradient arithmetic by computing Gradient for components and is arranged in matrix form to give Jacobian value. The resulting derivative evaluation uses the operator overloading concept which uses computer programs written in $\mathrm{C}++$.


Keywords: Automatic Differentiation, Augmented algebra, operator overloading, Forward mode

## 1. Introduction

Any efficient non-linear optimization routine needs good gradient approximations. Over the last decades, several research groups have developed the technique of Automatic Differentiation, which generates exact derivatives for a given code segment. A comprehensive introduction to this method can be found in (Griewank, A., 2000 \& 1990; Naumann, U., 2008; Moore, R.E., 1962; Rall, L.B., 2007). Automatic Differentiation can be implemented in various ways, each of which is dependent on circumstances partially. Here, we use a new methodology to implement AD for computing Gradient, Hessian and Jacobian. A new arithmetic is obtained based on the chain rule and using augmented algebra of real numbers through forward mode of Automatic Differentiation.

## 2. The Differentiation Arithmetic for Evaluation of Gradient and Hessian

To obtain the arithmetic for Gradients and Hessians, an ordered triples of the form $U=\left(u_{f}, u_{g}, u_{h}\right)$ with $u_{f} \in \mathbb{R}, u_{g} \in \mathbb{R}^{n}, u_{h} \in \mathbb{R}^{n} \times \mathbb{R}^{n}$ where the scalar $u_{f}$ denotes the function value $u(x)$ of the twice differentiable function $u: \mathbb{R}^{n} \rightarrow \mathbb{R}$, the vector $u_{g}$ and the matrix $u_{h}$ denote the value of the gradient $\nabla u(x)$ and the Hessian $\nabla^{2} u(x)$ respectively, each at a fixed point $x \in \mathbb{R}^{n}$ is considered. For the constant function $u(x)=c$, the triple is $U=\left(u_{f}, u_{g}, u_{h}\right)=(c, 0,0)$.For the function $u(x)=x_{k}$ with $k \in\{1,2, \cdots, n\},\left(u_{f}, u_{g}, u_{h}\right)=\left(x_{k}, e^{(k)}, 0\right)$ is
defined, where $e^{k} \in \mathbb{R}^{n}$ denotes the $k^{t h}$ unit vector and 0 denotes the zero matrix, respectively. There are some rules for the differentiation arithmetic similar to one dimensional case. They are

$$
\begin{aligned}
& W=U+V \Rightarrow\left(\begin{array}{l}
w_{f}=u_{f}+v_{f}, \\
w_{g}=u_{g}+v_{g}, \\
w_{h}=u_{h}+v_{h}
\end{array}\right) \\
& W=U-V \Rightarrow\left(\begin{array}{c}
w_{f}=u_{f}-v_{f}, \\
w_{g}=u_{g}-v_{g}, \\
w_{h}=u_{h}-v_{h}
\end{array}\right) \\
& W=U \cdot V \Rightarrow\left(\begin{array}{c}
w_{f}=u_{f} \cdot v_{f}, \\
w_{g}=u_{g} v_{f}+u_{f} v_{g}, \\
w_{h}=v_{f} \cdot u_{h}+u_{g} \cdot v_{g}^{T}+v_{g} \cdot u_{g}^{T}+u_{f} \cdot v_{h}
\end{array}\right) \\
& w_{f}=u_{f} / v_{f}, \\
& w_{g}=\left(u_{g}-w_{f} \cdot v_{g}\right) / v_{f}, \\
& W=U / V \Rightarrow\left(\begin{array}{c} 
\\
w_{h}=\left(u_{h}-w_{g} \cdot v_{g}^{T}-v_{g} \cdot w_{g}^{T}-w_{f} \cdot v_{h}\right) / v_{f}
\end{array}\right)
\end{aligned}
$$

where the familiar rules of calculus have been used in second and third components, and $v_{f} \neq 0$ is assumed in the case of division. The operations for $w_{f}, w_{g}$ and $w_{h}$ in these definitions are operations on real numbers, vectors and matrices. If the independent variables $x_{i}$ of a formula for a function $f: \mathbb{R}^{n} \rightarrow \mathbb{R}$ and $x \rightarrow f(x)$ are replaced by $X_{i}=\left(x_{i}, e^{(i)}, 0\right)$, and if all constants $c$ are replaced by their $(c, 0,0)$ representation, then evaluation of $f$ using the rules of differentiation arithmetic gives

$$
f(X)=f\left(\begin{array}{c}
X_{1} \\
X_{2} \\
X_{3} \\
\cdot \\
\cdot \\
\cdot \\
X_{n}
\end{array}\right)=f\left(\begin{array}{c}
\left(x_{1}, e^{(1)}, 0\right) \\
\left(x_{2}, e^{(2)}, 0\right) \\
\left(x_{3}, e^{(3)}, 0\right) \\
\cdot \\
\cdot \\
\cdot \\
\left(x_{n}, e^{(n)}, 0\right)
\end{array}\right)=\left(f(x), \nabla u(x), \nabla^{2} u(x)\right)
$$

For the elementary function $S: \mathbb{R} \rightarrow \mathbb{R}$ and $U=\left(u_{f}, u_{g}, u_{h}\right)$, the differentiation arithmetic is

$$
W=S(U) \Rightarrow\left(\begin{array}{c}
w_{f}=S\left(u_{f}\right), \\
w_{g}=S^{\prime}\left(u_{f}\right) \cdot u_{g}, \\
w_{h}=S^{\prime \prime}\left(u_{f}\right) \cdot u_{g} \cdot u_{g}^{T}+S^{\prime}\left(u_{f}\right) \cdot u_{h}
\end{array}\right)
$$

### 2.1 Algorithmic Description

Here the algorithm for the elementary operators $\{+,-, \cdot, /\}$ and for elementary functions $S \in\{$ power, $\exp , \log$, $\sin , \cos , \tan , \cot$, asin, acos, atan, acot, sinh, cosh, tanh, coth, asinh, acosh, atanh, acoth\} of a Gradient and Hessian arithmetic for a twice continuously differentiable function $f: \mathbb{R}^{n} \rightarrow \mathbb{R}$ is discussed.
Algorithm for overloading the $\pm$ operator for $U$ and $V$ is given by

Step 1: $\quad\left[W_{f}\right]=\left[u_{f}\right] \pm\left[v_{f}\right] ; \quad$ (Function value)
Step 2: for $i=1: n$
$\left[W_{g}\right]_{i}=\left[u_{g}\right]_{i} \pm\left[v_{g}\right]_{i} ; \quad$ (Gradient component
for $j=1: i$
$\left[W_{h}\right]_{i j}=\left[u_{h}\right]_{i j} \pm\left[v_{h}\right]_{i j} ; \quad$ (Hessian component)
Step 3: return $[W]$;

The algorithmic description for the multiplication operator for $(U, V)$ is given by

Step 1: $\quad\left[W_{f}\right]=\left[u_{f}\right] \cdot\left[v_{f}\right]$;
(Function value)
Step 2: for $i=1: n$

$$
\begin{array}{ll}
{\left[W_{g}\right]_{i}=\left[v_{f}\right] \cdot\left[u_{g}\right]_{i}+\left[u_{f}\right] \cdot\left[v_{g}\right]_{i} ;} & \text { (Gradient component } \\
\text { for } j=1: i \\
{\left[W_{h}\right]_{i j}=\left[v_{f}\right] \cdot\left[u_{h}\right]_{i j}+\left[u_{g}\right]_{i} \cdot\left[v_{g}\right]_{j}+\left[v_{g}\right]_{i} \cdot\left[u_{g}\right]_{j}+\left[u_{f}\right] \cdot\left[v_{h}\right]_{i j} ;} & \text { (Hessian component) }
\end{array}
$$

Step 3: return $[W]$;
The algorithm for overloading the / operator for $(U, V)$ is

```
Step 1: \(\quad\left[W_{f}\right]=\left[u_{f}\right] /\left[v_{f}\right]\)
(Function value)
Step 2: for \(i=1: n\)
    \(\left[W_{g}\right]_{i}=\left(\left[u_{g}\right]_{i}-\left[W_{f}\right] \cdot\left[v_{g}\right]_{i}\right) /\left[v_{f}\right] ; \quad \quad\) (Gradient component)
    for \(j=1: i\)
    \(\left[W_{h}\right]_{i j}=\left(\left[u_{h}\right]_{i j}-\left[W_{g}\right]_{i} \cdot\left[v_{g}\right]_{j}-\left[v_{g}\right]_{i} \cdot\left[W_{g}\right]_{j}-\left[W_{f}\right] \cdot\left[v_{h}\right]_{i j}\right) /\left[v_{f}\right] \quad\) (Hessian component)
```

Step 3: return $[W]$;
The algorithm for overloading the elementary functions for $(U, V)$ is

$$
\begin{array}{lll}
\text { Step 1: } & {\left[W_{f}\right]=S\left[u_{f}\right] ;} & \\
\text { Step 2: } & \text { for } i=1: n & \text { (Function value) } \\
& {\left[W_{g}\right]_{i}=S^{\prime}\left(\left[u_{f}\right]\right) \cdot\left[u_{g}\right]_{i} ;} & \\
& \text { for } j=1: i & \\
& {\left[W_{h}\right]_{i j}=S^{\prime \prime}\left(\left[u_{f}\right]\right) \cdot\left[u_{g}\right]_{i} \cdot\left[u_{g}\right]_{j}+S^{\prime}\left(\left[u_{f}\right]\right) ;} & \text { (Hesadient component) } \\
\text { Step 3: } & \text { return }[W] ; &
\end{array}
$$

Here, $W$ returns three components of which the first one is the function value, the second one is the Gradient and the third one is the Hessian.
INPUT: (i) Multi-variate functions of dimension 3.
(ii) Values of the components $x, y, z$

OUTPUT: (i) Value of the function
(ii) Value of the Gradient
(iii) Value of the Hessian

### 2.2 Numerical Results

The input function is $f(x)=x \sin (x)+\cos \left(y^{2}\right)+z^{2}$. The function is evaluated at the point $(2 \cdot 5,3 \cdot 5,4 \cdot 5)$
Value of the function $=523 \cdot 6875$
Value of the Gradient $=\left(\begin{array}{c}210 \cdot 875 \\ 220 \cdot 5 \\ 149 \cdot 625\end{array}\right)$
Value of the Hessian $=\left(\begin{array}{ccc}42 & 91 \cdot 45 & 49 \\ 91 \cdot 75 & 40 \cdot 5 & 42 \cdot 75 \\ 49 & 42 \cdot 75 & 31 \cdot 5\end{array}\right)$

## 3. Evaluation of Jacobians

Jacobian can be evaluated using Gradient arithmetic by computing Gradient for components $f_{i}$ with $i=$ $1,2, \cdots, n$. The same differentiation arithmetic, which is used for finding the functional value along with the Gradient, is used. Then, the Gradients for each $f_{i}, i=1,2, \cdots, n$ are computed and are arranged in the matrix form to give the Jacobian value. Using this new technique the Jacobian is computed exactly with minimum human effort.

## 4. Conclusion

Here a new technique is used for implementing AD for a scalar valued and twice continuously differentiable function $f: \mathbb{R}^{n} \rightarrow \mathbb{R}$. Automatic Differentiation is a useful tool as it facilitates the automatic generation of

Gradient and Hessian which enhances the robustness of an optimization algorithm that requires derivatives. The chain rule based differentiation arithmetic is used to find the Gradient and Hessian. Jacobian is also evaluated using Gradient arithmetic by computing Gradient for components and is arranged in the matrix form to give the Jacobian value.

## References

Griewank, A.(2000). Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation, SIAM, Philadelphia.
Griewank, A. (1990). 'Direct calculation of Newton steps without accumulating Jacobians', In: Coleman T.F. and Yuying Li (eds.), Large-Scale Numerical Optimization, SIAM, Philadelphia, Penn, pp. 115-137.
Naumann, U. and Uwe (2008). Optimal Jacobian accumulation is NP-complete, Mathematical Programming, Vol. 112, No. 2, pp. 427-441.
Moore, R.E. (1962). Interval Arithmetic and Automatic Error Analysis in Digital Computing', Ph.D. thesis, Department of Mathematics, Stanford University.
Rall, L.B. (2007), 'Early Automatic Differentiation: The Ch'in-Horner Algorithm', Reliable Computing, Springer, Vol. 13, pp. 303-308.

