Fuzzy Algorithmic Differentiation

Jonathan Hüser, Uwe Naumann

September 25, 2015

Abstract

Fuzzy arithmetic on fuzzy numbers can be used to introduce a form of uncertainty quantification into numerical computation. For a function $f : S^n \to S$ on fuzzy numbers at least two different kinds of sensitivities can be considered.

1. The sensitivities of the *parametrization* of the output fuzzy number $y = f(x)$ with regard to the *parametrization* of the input fuzzy numbers x_i for $1 \leq i \leq n$ (e.g. if $x_1 = (x_1^1, x_1^2, x_1^3)$ and $y = (y^1, y^2, y^3)$ are the parameters of triangular fuzzy numbers we can compute $\partial y^1 / \partial x_1^2$).
2. The derivatives of the fuzzy output with regard to the fuzzy inputs $\partial y / \partial x_i$ for $1 \leq i \leq n$ which again are fuzzy numbers.

We demonstrate how to apply adjoint algorithmic differentiation (AAD) via operator overloading to a fuzzy arithmetic via operator overloading to arrive at both kinds of sensitivities depending on the order of overloading. The application of AAD comes with its usual benefit of computing sensitivities for all inputs with regard to an output in $O(1) \cdot \text{cost}(f)$ instead of $O(n) \cdot \text{cost}(f)$ where $\text{cost}(f)$ is the computational cost of evaluating f.

1 Fuzzy Arithmetic using α-cuts

A fuzzy number x has a membership function μ_x which for each value $z \in \mathbb{R}$ assigns a degree of membership $\mu_x(z) \in [0, 1]$. There exists a value $\hat{z} \in \mathbb{R}$ for which $\mu_x(\hat{z}) = 1$ and μ_x is monotone around this value, i.e. μ_x is increasing left of $\hat{z}$ and decreasing right of $\hat{z}$ (for all $z_1 \leq z_2 \leq \hat{z}$ we have $\mu_x(z_1) \leq \mu_x(z_2)$ and for all $\hat{z} \leq z_1 \leq z_2$ we have $\mu_x(z_1) \geq \mu_x(z_2)$).

An α-cut of a fuzzy number x is the set of numbers $z \in \mathbb{R}$ with a membership $\mu_x(z) \geq \alpha$. Because of the monotony of μ_x around $\hat{z}$ an α-cut can be described by an interval $[z_1, z_2]$ with $z_1 \leq \hat{z} \leq z_2$ and $\mu_x(z_1) = \mu_x(z_2) = \alpha$. A fuzzy number can be described by sufficiently many α-cuts depending on the complexity of its membership function. Fuzzy arithmetic operations are carried out by performing interval arithmetic on the α-cuts of the same levels.

Example Consider the addition of fuzzy numbers $y = x_1 + x_2$ with α-cuts being defined by the following intervals

α	x_1	x_2	y
1	$\{5\}$	$[9, 11]$	$[14, 16]$
0.5	$[4, 6]$	$[5, 15]$	$[9, 21]$
0	$[0, 10]$	$[4, 16]$	$[4, 26]$

Fuzzy arithmetic can be implemented via operator overloading by providing a data type that contains an array of intervals describing the α-cuts. Binary operators are overloaded by applying the interval arithmetic operator for each pair of same level α-cuts of both operands.

Given an interval type

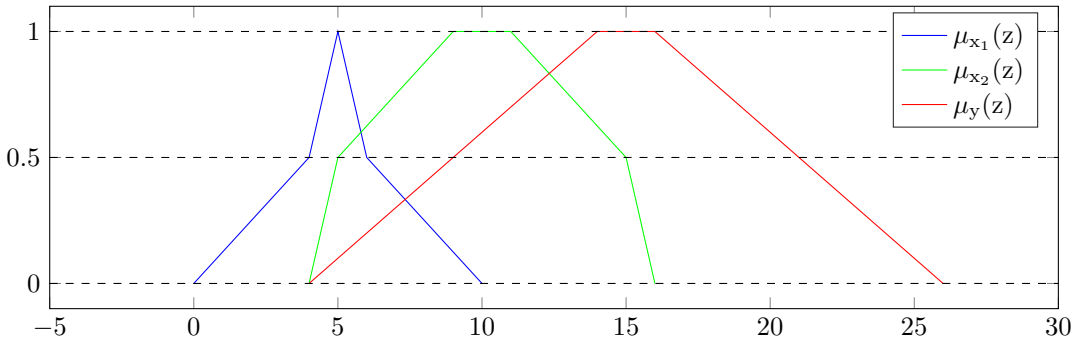

Figure 1: Fuzzy addition $y = x_1 + x_2$

```
template<class T> class interval {
        private:
                T low;
                T up;
};
```

we can implement a fuzzy type

```
template<class T, int N> class fuzzy {
        private:
                interval<T> cut[N];
}
```

We can instantiate the fuzzy data type with 2 α-cuts as follows

```
fuzzy<double, 2> x;
```

A binary operator like addition is overloaded as follows

```
template<class T, int N> inline
fuzzy<T, N> operator+(fuzzy<T, N> const & x, fuzzy<T, N> const & y) {
        fuzzy<T, N> temp;
        for (int i = 0; i < N; i++)
                temp.set_cut(i, x.get_cut(i) + y.get_cut(i));
        return temp;
}
```

where the addition of two α-cuts is implemented via an interval arithmetic library.

2 Combining AD and Fuzzy Arithmetic

We can instantiate the `dco/c++` adjoint data type of with base type `double` as follows

```
dco::ga1s<double>::type x;
```

There are two ways of combining the fuzzy type with the dco type

1. A fuzzy type with underlying adjoint data type can be used, which results in adjoint data types for interval bounds of all α-cuts.

   ```
   fuzzy<dco::ga1s<double>::type, 2> x;
   ```

   Using this approach we can compute sensitivities of the parametrization of the fuzzy number that is the function output with regard to the parametrizations of the fuzzy numbers that are the inputs of the function.

2. An adjoint type with underlying fuzzy data type can be used, which results in the computation of the chain rule and partial derivatives using fuzzy data types and the sensitivities thus being fuzzy values.

```
dco::ga1s< fuzzy<double, 2> >::type x;
```

   Differentiation rules for fuzzy numbers need to be implemented into the AD framework in order to arrive at the desired derivative definitions.

3 Speelpenning Example

An example of the two approaches introduced in the previous section using the Speelpenning function $y = \prod_{2=i}^{N} x_i$ can be downloaded from http://www.stce.rwth-aachen.de/software/fuzzyAD.html.

In the `fuzzyAD.cpp` file consider the following four functions:

- `_dco()` illustrates the use of the `dco/c++` adjoint data type and tape for AAD of the speelpenning function.
- `_fuzzy()` illustrates the use of the fuzzy data type as is and computes the sensitivity with regard to one α-cut interval bound using a finite difference approach.
- `_dco_fuzzy()` instantiates the adjoint data type with a fuzzy base type, i.e. we compute the adjoint sensitivities $x_{i,(1)}$ as fuzzy numbers.
- `_fuzzy_dco()` instantiates the fuzzy data type with an adjoint base type and illustrates how to compute adjoint sensitivities of individual α-cut interval bounds.

In the `fuzzy_uncertainty.cpp` file we compare three different approaches of uncertainty quantification for the Speelpenning function:

- Fuzzy sets with triangular form as described above. See `_fuzzy()`
- Monte Carlo where we sample inputs from a triangular distribution. See `_monte_carlo()`
- First order moments propagation, i.e. approximating variance by the first order Taylor expansion, with the assumption that the output also follows a symmetric triangular distribution (which it does not). See `_dco_uncertainty()` and `_fd_uncertainty()`.

Since fuzzy set theory is not based on probability theory it is not really "fair" to compare a fuzzy set approach with a probabilistic approach.

In Figure 2 we show the input distributions or fuzzy sets for inputs with mean 1 and variance 0.01 (for fuzzy set we use variance as 0-cut). We normalize in such a way that the probability density function has a value of 1 at the tip of the triangle. With the normalization the triangular fuzzy set and distributions look the same.

The output uncertainty is quantified by the output distributions or fuzzy set in Figure 3. We see that the fuzzy set gives a strong "over approximation" of the uncertainty even for small variance. The higher moments of the distribution are clearly seen missing from the moments based approach.

Note that both a fuzzy set and first order moments based approach can be accomplished with $O(1) \cdot \text{cost}(f)$ (using an adjoints) but Monte Carlo costs is $n \cdot \text{cost}(f)$ where n is the number of samples required in terms of limit theorems.

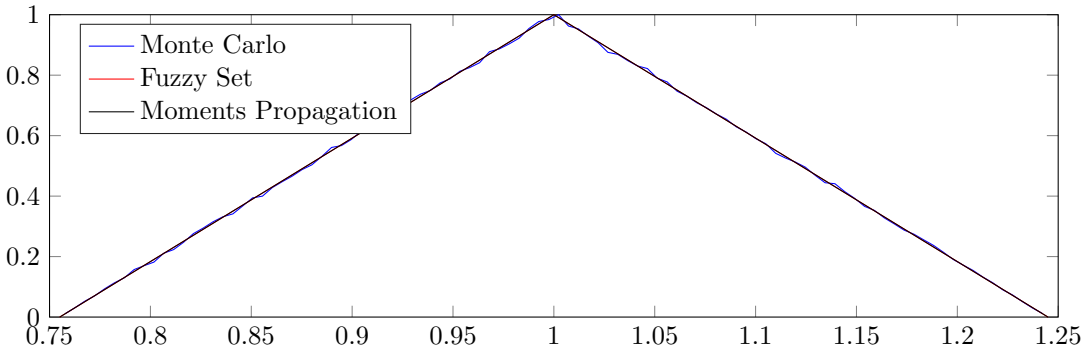

Figure 2: Normalized distributions and fuzzy set for inputs

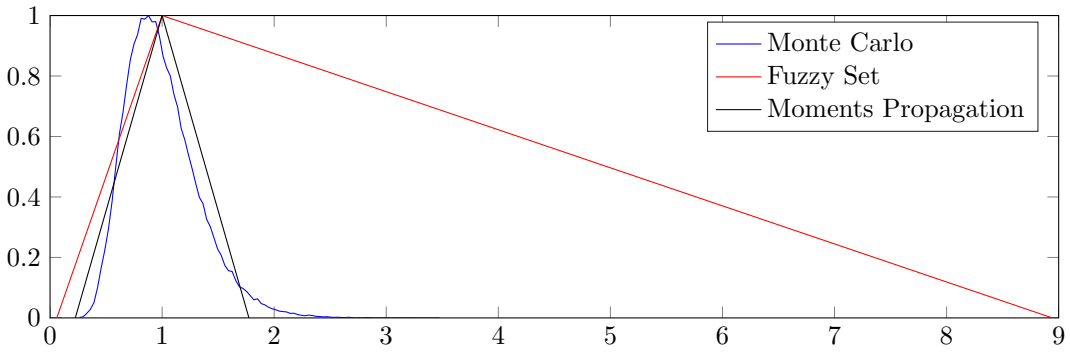

Figure 3: Normalized distributions and fuzzy set for outputs