Miller Analyser for Matlab — A Matlab Package for Automatic Roundoff Analysis

Attila Gáti

Miller Analyser for MATLAB is an automatic roundoff error analyser software, that extends the work of Miller et al. [1, 2, 3]. The software runs within the MATLAB environment, and can test the stability of numerical methods given as m-functions. Based on the algorithm of Miller, a number $\omega(d)$ is associated with each set d of input data. The function $\omega(d)$ measures rounding error, i.e. $\omega(d)$ is large exactly when the method applied to d produces results which are excessively sensitive to rounding errors. A numerical maximizer is applied to search for large values of ω. Finding large values of ω can be interpreted, that the given numerical method is suffering from a specific kind of instability.

We can perform analysis based on several error measuring numbers (various ways of assigning ω), and beside analysing the propagation of rounding errors in a single algorithm we can also compare the numerical stability of two competing numerical methods, which neglecting rounding errors compute the same values.

The analysis is based on the standard model of floating point arithmetic, which assumes that the individual relative rounding errors on arithmetic operations are bounded by the machine rounding unit. Practically, the computed result equals the correctly rounded exact result. The IEEE 754/1985 standard of floating point arithmetic guarantees that the standard model holds for addition, subtraction, multiplication, division and square root. Unfortunately, it is not true for the exponential, trigonometrical, hyperbolical functions and their inverses. Hence, we can analyse only numerical algorithms that can be decomposed to the above mentioned five basic operations and unary minus, which is considered error-free.

The first step of computing the error function $\omega(d)$ is building the computational graph of the analysed numerical method. Decomposition of a numerical method at a particular input $d = d_0$ to the allowed arithmetic operations give rise to a directed acyclic graph, the computational graph, with a node for each input value, output value and operation. There are arcs from each arithmetic node (ie., one corresponding to an operation) to the nodes for its operands and from each output node to the operation that computes its value.

According to the resulting computational graph the output computed as a function: $R_{d_0}(d,\delta)$ ($R_{d_0} : \mathbb{R}^{n+m} \to \mathbb{R}^k$), where $d \in \mathbb{R}^n$ is the input vector, and δ is the vector of individual relative rounding errors on the m arithmetic operations ($\delta \in \mathbb{R}^m$, $\|\delta\|_\infty \le u$, where u is the machine rounding unit). The computation of $\omega(d)$ is based on the partial derivatives of R_{d_0} with respect to the input and the rounding errors. We apply automatic differentiation on the graph in reverse order, ie. the chain rule is applied in the opposite direction as the basic operations were executed.

References

[1] Webb Miller, *Software for roundoff analysis*, ACM Trans. Math. Softw. 1 (1975), no. 2, 108–128.

[2] Webb Miller and David Spooner, *Software for roundoff analysis*, ii, ACM Trans. Math. Softw. 4 (1978), no. 4, 369–387.

[3] Webb. Miller and Celia Wrathall, *Software for roundoff analysis of matrix algorithms*, Academic Press, New York, 1980