Reading and Thinking

SIAM J. SCI. COMPUT.
Vol. 39, No. 2, pp. C116–C143

IS THE MULTIGRID METHOD FAULT TOLERANT? THE TWO-GRID CASE*

MARK AINSWORTH† AND CHRISTIAN GLUSA‡

This paper is dedicated to Professor Ivo Babuška on the occasion of his 90th birthday

Abstract. The predicted reduced resiliency of next-generation high performance computers means that it will become necessary to take into account the effects of randomly occurring faults on numerical methods. Further, in the event of a hard fault occurring, a decision has to be made as to what remedial action should be taken in order to resume the execution of the algorithm. The action that is chosen can have a dramatic effect on the performance and characteristics of the scheme. Ideally, the resulting algorithm should be subjected to the same kind of mathematical analysis that was applied to the original, deterministic variant. The purpose of this work is to provide the first rigorous analysis of the behavior of the multigrid algorithm in the presence of faults. Specifically, we prove estimates on the behavior of the Two Grid Method similar to the classical asymptotic results. Multigrid is arguably the method of choice for the solution of large-scale linear algebra problems arising from discretization of partial differential equations, and it is of considerable importance to anticipate its behavior on an exascale machine. The analysis of resilience of algorithms is in its infancy, and the current work is perhaps the first to provide a mathematical model for faults and analyze the behavior of a state-of-the-art algorithm under the model. It is shown that the Two Grid Method fails to be resilient to faults. Attention is then turned to identifying the minimal necessary remedial action required to restore the rate of convergence to that enjoyed by the ideal fault-free method.

Key words. multigrid, fault tolerance, resilience, random matrices, convergence analysis

AMS subject classifications. 65F10, 65N22, 65N55, 68M15

DOI. 10.1137/16M1100691

- Have you experienced errors when you program? Of course, you did.
- What types of errors you have made?
- Any of the errors not made by yourselves? What are they?
- How can you make your code more reliable?
- How can you make your algorithms more fault-tolerant? Any plans?