8. C. Segre, *Intorno alla storia del principio di correspondenza e dei sistemi di curve*, Bibl. Math. **6** (1982), 33–47 = *Opere*, Vol. 1, Edizioni Cremonese, Rome, 1957, pp. 185–197.

9. H. G. Zeuthen and M. Pieri, *Géométrie enumerative*, Encyclopédie des Sciences Mathématiques, III.4, Gauthier-Villars, Paris, 1915.

STEVEN L. KLEIMAN

BULLETIN (New Series) OF THE
AMERICAN MATHEMATICAL SOCIETY
Volume 12, Number 1, January 1985

0025-5718/85 $1.00 + $.25 per page

*Mathematical experiments on the computer*, Academic Press, New York, 1982, xvii + 525 pp., $39.50. ISBN 0-1230-1750-5

Mathematics is ancient, the computer is new. Both involve the expenditure of energy to manipulate symbols and thereby create structure. At the very least they complement each other, though they often become corrupting influences. Hilbert hoped that mathematics could be managed by computation; Godel proved that it couldn't. In some quarters limitations on the management of computation, e.g., program verification, by mathematics is taken as a measure of the immaturity of our understanding of computation. Rejecting those august heights of mutual absorption, we find that a pragmatic use of the one by the other is healthy for both.

This book deals with one such use: the computer as a calculator to lend credence to conjectures that could be fragments of theorems or suggestive of theorems. The conjectures have in common a significant computational component. None involves the use of the computer as primarily a manipulator of formuale or logical truths. The author is careful to point out that he is considering experiments that are driven by numerical computation. In a more perfect world an experiment on the computer could invoke both the formula manipulation world exemplified by MACSYMA [**1**] and that recommended by the author. Unfortunately the world of the former speaks LISP [**2**] and the latter speaks APL [**3**] and the two in each other's presence are tongue-tied. Some day....

Conjectures of any kind tend to be plastic and tentative. The programming language and computing environment used in exploration should have the same behavior—programs should be easy to write, test, generalize, and discard. Even more, they should be terse and support a great deal of implicit logical and mathematical structure that would otherwise need to be made explicit by programming. The author's choice of APL is to be applauded. The APL environment, its workspace and library, the terseness of its programs, and the ease with which APL programs can be tested and modified make it the best available language for the purposes Grenander has in mind. Unfortunately there is a viper in this Garden of Eden: Learning to program well in APL is considerably more difficult than in any other language. Even though the mastery pays big dividends, most casual users are unwilling to make the required initial time investment. Other programming languages such as BASIC