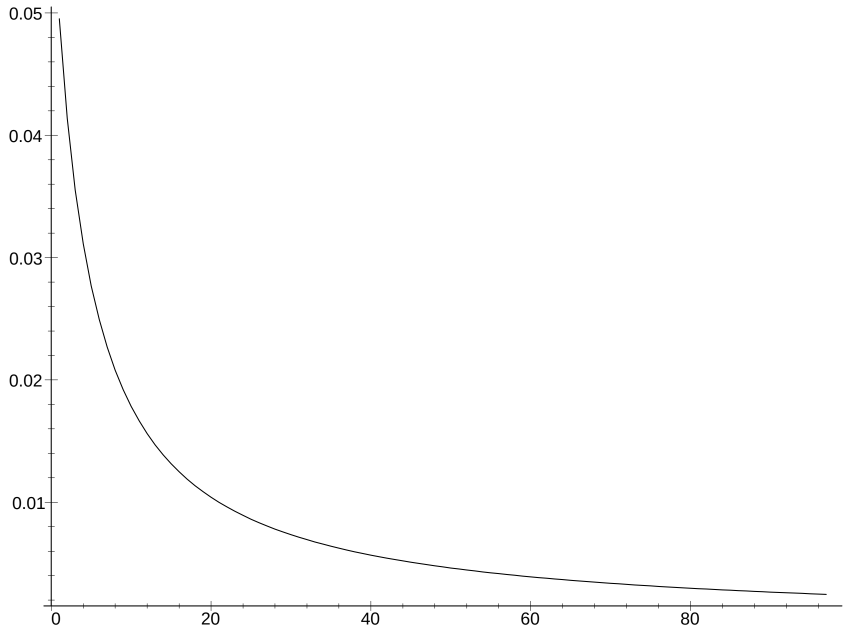

Figure 0.3. A plot of 100 terms of the absolute error α_n.

$$\rho_n = \frac{|\pi/4 - P(n)|}{\pi/4} = \frac{\alpha_n}{\pi/4}.$$

The *relative* error function ρ is plotted in Figure 0.4. Notice that each error function tends toward zero. Numerical analysts are concerned with the *rate* at which the error goes to zero (with respect to terms in a series or iterations of a numerical technique). All other things being equal, approximations with errors that converge more quickly to zero are considered superior. We shall have more to say about error later on in this book.

Exercise 0.2. What could be meant by the statement ‘‘all other things being equal’’? How else could different approximations to the same function or value actually be different?

It is worth reviewing our philosophy. What new techniques have we been using to explore our numerical approximation for π? First, we have been using an *interactive* environment that allows us the flexibility to explore new ideas and modify them quickly. Second, graphical information display plays an important role. Third, symbolic representations allow us to explore concepts more directly.

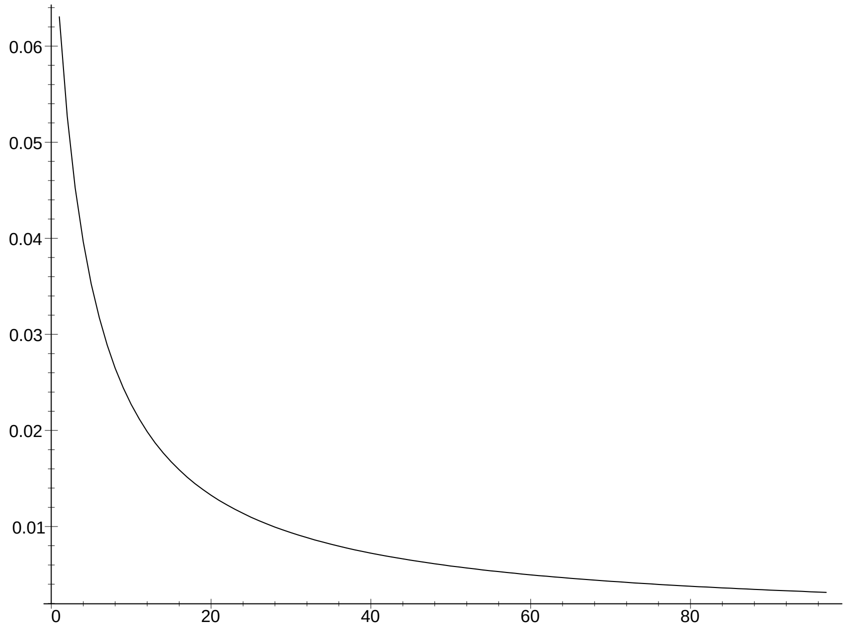

Figure 0.4. A plot of 100 terms of the relative error ρ_n.

The use of symbolic computation is not a magic solution to our problems. For example, we have only *qualitative* evidence that if our goal is to compute π efficiently, then Leibnitz' formula may not be directly practical. We do not have a proof of the convergence rate, nor will an efficient technique be automatically found for us.

As we progress through this book, we shall see that symbolic and numeric computation are mutually supportive. Despite being quite powerful for some classes of computation, symbolic languages have their own idiosyncrasies and frustrations, and they can execute slowly by the standards of numerical methods implemented using traditional compiled programming languages. However, different purposes are served by numeric and symbolic approaches.

In computer science, we measure the *efficiency* of an algorithm or an implementation by proving properties of its running time and memory usage. A good implementation of any algorithm, and especially a numerical method, is subject to mathematical scrutiny before its implementation, and careful empirical testing afterward. When performing interactive mathematical computations, notions of space and time efficiency usage are fuzzier, because the way in which the system performs many of its functions is less clear than in a conventional programming language. There is no guarantee that a certain statement will execute quickly, but in the initial stages of our enquiry, we may be satisfied to