What is more probable, flipping 10 coins and getting no heads or flipping 10,000 coins and getting exactly 5000 heads? [Pencil]

22. The double factorial is defined as

$$n!! \equiv n \times (n-2) \times (n-4) \ldots \times \begin{cases} 6 \times 4 \times 2 & n \text{ even} \\ 5 \times 3 \times 1 & n \text{ odd} \end{cases}$$

(a) Write a program that prints out $n!!$ by evaluating its definition using logarithms. Test your program by checking that $1000!! \approx 3.99 \times 10^{1284}$; compute 2001!!. (b) Obtain an expression for $n!!$ in terms of $n!$. [Pencil] (c) Write a program that prints out $n!!$ by evaluating it using Stirling's approximation when $n > 30$. Compute 10000!!, 314159!! and $(6.02 \times 10^{23})!!$. [Computer]

23. Suppose that you are standing at 40° latitude and are 2 m tall. (a) Find the velocity of your feet, v, due to the rotation of Earth. Assume that Earth is a perfect sphere of radius $R = 6378$ km and that a day is exactly 24 h long. (b) Using the result from (a), compute the centripetal acceleration at your feet as $a = v^2/r$. (c) Repeat parts (a) and (b) for your head, and compute the difference between the acceleration at your head and feet. Show how round-off can corrupt your calculation and how to fix the problem. [Pencil]

24. (a) Write a program to reproduce Figure 1.3. Use $h = 1,\ 10^{-1},\ 10^{-2}, \ldots$. (b) Modify your program to use $h = 1,\ 2^{-1},\ 2^{-2}, \ldots$. The results are strikingly different; explain why. [Computer]

25. Write a program to find ϵ_r, the smallest number that when added to 1 returns a value different from 1. Compare your result with either MATLAB's built-in function `eps` or `DBL_EPSILON`, as defined in the C++ `<float.h>` header. [Computer]

26. Consider the Taylor expansion for the exponential

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots = \lim_{N\to\infty} S(x, N)$$

where $S(x, N)$ is the partial sum with $N+1$ terms. (a) Write a program that plots the absolute fractional error of the sum, $|S(x, N) - e^x|/e^x$, versus N (up to $N = 60$) for a given value of x. Test your program for $x = 10, 2, -2$, and -10. From the plots, explain why this is not a good way to evaluate e^x when $x < 0$.[49] (b) Modify your program so that it uses the identity $e^x = 1/e^{-x} = 1/S(-x, \infty)$ to evaluate the exponential when x is negative. Explain why this approach works better. [Computer]

BEYOND THIS CHAPTER

The Lagrange formulation for assembling an interpolating polynomial has only one advantage: It is easy to understand and remember. Algorithmically, it is not the best method for polynomial interpolation for a variety of reasons, including susceptibility to round-off error when fitting higher-order polynomials. A superior approach is to build a divided difference table and use the Newtonian formulation.[33]

The round-off error in a numerical scheme can be quantitatively estimated using backward error analysis.[133] It is possible to perform calculations of arbitrary precision by storing numbers as rational quotients (fraction with integer