New York University
Courant Institute of Mathematical Sciences

G22.2112-001/G63.2043 Scientific Computing

Handout #2
Jan. 18, 2001

Yu Chen

Assignment 2

Objective: To explore computer arithmetic. Implement only the double precision part of the questions, unless you use a language (such as Fortran or C) where setup for single precision is easy.

1. The fibonacci numbers, f_k, are defined by $f_0 = 1$, $f_1 = 1$, and

$$f_{k+1} = f_k + f_{k-1} \tag{1}$$

for any integer $k > 1$. A small perturbation of them, the "pib numbers" ("p" instead of "f" to indicate either the pentium bug or perturbation), p_k, are defined by $p_0 = 1$, $p_1 = 1$, and

$$p_{k+1} = c \cdot p_k + p_{k-1} \tag{2}$$

for any integer $k > 1$, where $c = 1 + \sqrt{3}/100$.

a. Make a SEMILOGY plot of f_n and p_n as a function of n. On the plot, mark $1/\epsilon_{mach}$ for single and double precision IEEE floating point arithmetic. This can be useful in answering the questions below.

b. For various n values, compute the f_k for $k = 2, 3, \ldots, n$ using (1). Then rewrite (1) to express f_{k-1} in terms or f_k and f_{k+1}. Use the computed f_n and f_{n-1} to recompute f_k for $k = n-2, n-3, \ldots, 0$. Make a plot of the difference between the original $f_0 = 1$ and the recomputed f_0 as a function of n. What n values result in no accuracy for the recomputed f_0? How would the results in single and double precision differ?

c. Repeat b. for the pib numbers. Comment on the striking difference in the way precision is lost in these two cases. Which is more typical? *Extra credit*: predict the order of magnitude of the error in recomputing p_0 using what you may know about recurrence relations and what you should know about computer arithmetic.

2. The binomial coefficients, $a_{n,k}$ (in Matlab nchoosek(n,k)), are defined by

$$a_{n,k} = \begin{pmatrix} n \\ k \end{pmatrix} = \frac{n!}{k!(n-k)!} \tag{3}$$

To compute the $a_{n,k}$, for a given n, start with $a_{n,0} = 1$ and then use the recurrence relation $a_{n,k+1} = \frac{n-k}{k+1} a_{n,k}$.

(a) For a fixed of n, compute the $a_{n,k}$ this way, noting the largest $a_{n,k}$ and the accuracy with which $a_{n,n} = 1$ is computed. Do this in single and double precision. Why is it that roundoff is not a problem here as it was in problem (1)?

b. Use the algorithm of part (a) to compute (we think of k as a random variable, the number of heads in n tosses of a fair coin, with $E(n)$ being the expected value of k)

$$E(n) = \frac{1}{2^n} \sum_{k=0}^{n} k a_{n,k} = \frac{n}{2} \quad . \tag{4}$$

Write a program without any safeguards against overflow or zero divide (*this time only!*)[1]. Show (both in single and double precision) that the computed answer has high accuracy

[1] One of the purposes of the IEEE floating point standard was to *allow* a program with overflow or zero divide to run and print results

as long as the intermediate results are within the range of floating point numbers. As with (a), explain how the computer gets an accurate, small, answer when the intermediate numbers have such a wide range of values. Why is cancellation not a problem? Note the advantage of a wider range of values: we can compute $E(n)$ for much larger n in double precision. Print $E(n)$ as computed by (4) and $M_n = \max_k a_{n,k}$. For large n (somewhere over 1000), one should be `inf` and the other `NaN`. Why?

c. For (fairly large) $n = 40$, plot $a_{n,k}$ as a function of $k = 1, 2, ldots, n$ to illuminate the interesting "bell shaped" behavior of the $a_{n,k}$ near their maximum.

d. (*Extra; extra credit, extra hard, don't spend too much time on it!*) Find a way to compute

$$S(k) = \sum_{k=0}^{n} (-1)^k \sin(2\pi \sin(k/n)) a_{n,k}$$

with good relative accuracy for large n.