Introduction to Floating-Point Computations

Douglas B. Meade

Department of Mathematics

Overview

Calculations with floating-point numbers must be done with care. Computations done without some forethought and planning can easily lead to incorrect or nonsensical conclusions. For example, based solely on numerical evidence, one might conclude the harmonic series converges. The purposes of this lab, and project, are to alert you to some of the potential problems and ways they can be addressed.

Preparation

The Activities and Project 3 involve various topics from earlier in the course:

- (conditional) convergence of the alternating harmonic series
- approximation of definite integrals using left, right, midpoint, trapezoid, and Simpson approximations.

Review these topics for this lab (and as early preparation for the final exam).

Maple Essentials

- New Maple commands introduced in this lab include:

Command	Description

`ApproximateInt`	visual, numeric, or symbolic representation of a Riemann sum;
	in the `Student[Calculus1]` package
	`> ApproximateInt(` *f*`,` *a*`..`*b*`, partition=`*N*`,`
	`> method=left, output=plot );`
	(see also Lab 13 for Calculus I, Fall 2004)

`Digits`	maximum number of significant (decimal) digits in a floating-point number `> Digits := 5;` (the default value is 10)
`print`	print one or more Maple quantities `> print( 9, 9^9, 9! );`
`printf`	formatted printing of one or more Maple quantities `> printf( "%5a %10a %10a\n", 9, 9^9, 9! );` (see `?printf` for full formatting details)

`[for ...] [from ...]`
`[to ... ] [by ...]`
`[while ...]`
`do ... end do`
	command for controlling iterative loops
	the `for`, `from`, `to`, and `by` clauses are all optional
	only the `do` and `end do` are required;
	explicit examples are provided in the Activities

Assignment

- Project 3 is due at the beginning of next week's lab.
- Mastery Quiz 10 will be distributed at the end of today's lab.
- In next week's lab you will have time to complete Mastery Quiz 11.

Activities

1. The following segment of Maple code uses a loop to find and display the first few Maclaurin coefficients of $f(x) = \arctan(x)$:

```
> f := arctan(x);                                  # define function
> a := 1;                                          # define base point for Taylor expansion
> for k from 1 to 15 do
>   df := diff( f, x$k );                          # compute k^th derivative
>   dfa := eval( df, x=a );                        # evaluated at x = 0
>   printf( " %3a %20a\n", k, dfa/k!  );           # display coefficient of (x-a)^k in Taylor polynomial
> end do:                                          # note use of colon here
```

2. Modify the code in Activity 1 to assemble the 20^{th} Maclaurin polynomial for $\operatorname{erf}(x) = \dfrac{2}{\sqrt{\pi}} \displaystyle\int_0^x e^{-t^2}\, dt$.

3. "Machine epsilon" is defined to be the largest positive number ϵ such that (numerically) $1 + \epsilon = 1$. (In the real numbers, $\epsilon = 0$, but not for floating-point numbers.) Consider the following Maple loop:

```
> Digits := 6;                                     # maximum number of decimal digits
> for k from 0 to 12 do
>  q := 1+10^(-k);                                 # compute 1 + ε                (†)
>  printf( " %3a %15a %30 %15a \n",
>          k, 10^k, q, evalf(q) );
> end do;
```

   An estimate of machine epsilon can be obtained from this table of values. How does machine epsilon change with the value of `Digits`? **Note:** *Maple has no trouble working with arbitrarily large integers; this effect is a concern only for floating-point arithmetic.*

4. Repeat Activity 3 with the command marked as (†) changed to `q := 1+10^k;`.

5. It is important to be aware of machine epsilon when trying to compute the value of an infinite series, $\sum_{k=1}^{\infty} a_k$. If the terms of the series converge to zero (as we know they must for any convergent series), then — eventually — the $a_k < \epsilon$ and adding additional terms contributes nothing to the computed sum. That is, the series appears to converge!

```
> a := k -> 1/k;                                   # define terms in harmonic series
> Digits := 6;                                     # maximum number of decimal digits
> SUM := 0.0;                                      # initialize sum to zero
> for k from 1 to 100000 do                        # add next term to sum         (‡)
>   SUM := SUM + a(k);
> end do:                                          # with a ;, output is 100,000 lines!
> SUM;                                             # final sum
```

   Change the value of `Digits` (and possibly the upper bound on the sum) and observe how the "sum" of the harmonic series changes.

6. Repeat Activity 5 for the alternating harmonic series. Explain why the sum of this series is not as sensitive to the value of `Digits`.

   Hint: *Recall that the alternating harmonic series is conditionally convergent.*

7. One way to avoid (or at least postpone) the effects of machine epsilon is to add numbers from the smallest to the largest. To reverse the order in which the terms are added in Activity 5, change the line marked as (‡) to

   `for k from 100000 to 1 by -1 do`.

   Repeat the calculations done in Activity 5 for different values of `Digits` and compare the sums with those obtained in Activity 5. **Note:** *This takes advantage of the fact that floating-point addition is not necessarily associative:* $(a + b) + c \neq a + (b + c)$.

Project 3: Convergence Rates for Numerical Integration

When you talked about Riemann sum approximations for definite integrals, you were told that the Riemann sums converge to the exact value of the definite integral as the maximum width of a subinterval in a partition decreases to zero. **This convergence is not seen numerically.** There are several factors involved in this breakdown, including the problems seen earlier in this lab with the computation of a sum that involves widely different scales of numbers.

For this project you will investigate the convergence rates for five different numerical methods — left, right, midpoint, trapezoid, and Simpson's rule — for approximating the definite integral $\displaystyle\int_{-1}^{2} \frac{2}{\sqrt{\pi}} e^{-t^2}\, dt$. **Note:** *This integral is related to the error function, introduced in Lab K.*

1. Each numerical method has a different rate of convergence. Assuming the error can be written in the form $E = An^{-p}$. where E is computed, n is the number of partitions (subintervals), A is a constant that can be computed (or bounded) based on the problem at hand, and p is the rate of convergence for this method. Suppose the test is exectuted twice: once with $N = n_1$ and error E_1 and once with $N = n_2$ and error E_2. Given the results of two computations, find a general formula for p that does *not* involve A.
   Note: *For your Project Report it is not necessary to give all details, but the major steps must be summarized: what equations do you solve? how do you eliminate A? ....*
2. The Maple code provided with this assignment creates a table of data showing the approximations and errors for a left endpoint Riemann sum with $N = 2$, 4, 8, 16, 32, 64, 128, 256, 512, and 1024 subintervals. Execute this code and report the results in a similar table that contains columns for the number of subintervals, the corresponding left Riemann sum, error with the exact value, and the convergence rate (as computed using your formula in 1.). You will need to modify the Maple code, or use another technology, to compute the convergence rate according to your formula. You should see that the errors decrease (slowly) towards zero. To what value do the convergence rates converge? **Note:** *Use* `Digits := 10;`.
3. Repeat 2. for the (i) right endpoint, (ii) midpoint, (iii) trapezoid, and (iv) Simpson's rules. (Use `Digits := 10` for 3., as well) In addition to presenting the tables of values, identify all cases where the p ceases to converge to the expected value.
4. For each of the five different types of Riemann sum approximations, find the smallest value of `Digits` for which the approximations continue to show the expected improvement for all values of N through 1024.
5. For each of the five different types of Riemann sum approximations, find the number of subintervals N and value for `Digits` needed to obtain an approximation that is correct to 8 decimal places to the right of the decimal point.