

MCS 471 Project One: Summing Sequences of Numbers

The goal of this project is to investigate numerical methods to sum sequences of numbers using floating-point arithmetic. To experiment with various precisions, we will use Maple. This document is processed from a Maple worksheet, available from the web site for this class. It is highly recommended to start the project by downloading this worksheet.

0. Summing floating-point numbers

In this project we consider numerical methods to sum sequences of floating-point numbers. We use two methods. The first method is the plain addition (prep school method). The second method first sorts the numbers before adding them. In Maple, these two methods are respectively provided in the procedures “plain_add” and “sort_add”.

```
[> plain_add := n -> add(k,k=n): # plain addition
[> sort_add:= n->add(k,k=sort(n)): # first sort then add
```

For example, we use these procedures as follows:

```
[> our_list := [1.0,9.0,0.7]:
[> plain_add(our_list);
                                10.7
[> sort_add(our_list);
                                10.7
```

The input of these Maple procedures is a list of numbers. A list in Maple is a sequence of elements, separated by commas and enclosed by square brackets.

For integer numbers, the two methods will always return the same result. Working with floating-point arithmetic, the summing of long lists will give different results. To compare our numerical results with the exact sum, we avoid roundoff errors by converting first to rational numbers:

```
[> exact_add := n -> plain_add(map(x->convert(x,rational),n)):
```

Then the exact sum is also a rational number:

```
[> s := exact_add(our_list);
                                107
s := ---
                                10
```

which we convert to a floating-point number with evalf:

```
[> evalf(s);
                                10.70000000
```

As the problem of summing numbers is so simple, lots of real-world applications can be found. For example, suppose we monitor the stock prize as it is going up and down almost continuously during the day. At the end of the day we want to sum up all the increments and decrements to know the final value of the sum. A similar calculation must be made at a cash register in a store, adding up the money going in and the money going out.

1. Summing random sequences

We first generate a random sequence of uniformly distributed numbers.

```
[> randomize(): # initialize the seed
[> random_data := [stats[random,uniform[-1,1]](5)]:
```

We just generated 5 random numbers, uniformly distributed between -1 and 1 . For longer lists, we may want to suppress the output, replacing the semicolon at the end of the command by a colon.

```
[> exact_sum := exact_add(random_data):
```

To get a floating-point approximation, we use the `evalf` command:

```
[> approx_sum := evalf(exact_sum);
      approx_sum := -1.019136378
```

Maple converts to its default precision of 10 digits. To see the error made, we use `evalf` with as extra argument the number of digits we wish to calculate with:

```
[> evalf(exact_sum-approx_sum,20);
      -9
      -.4009374938 10
```

The magnitude of the error equals the default precision of Maple. We can restrict the precision to 3 digits

```
[> Digits := 3:
[> plain_sum := plain_add(random_data);
      plain_sum := -1.01
[> sort_sum := sort_add(random_data);
      sort_sum := -1.02
```

Already for such as small sequence we sometimes observe a significant difference in the errors:

```
[> error_of_plain_add := abs(evalf(exact_sum-plain_sum,8));
      error_of_plain_add := .0091364
[> error_of_sort_add:= abs(evalf(exact_sum-sort_sum,8));
      error_of_sort_add := .0008636
```

Observe that we compute the error with higher precision than the precision used to obtain the sums.

Assignment One

Repeat the little experiment above with longer lists of numbers and for various levels of precision. Take lists of length 100, 200, 300,... and calculate the sums with values for `Digits` equal to 20, 30, 40,..., each time comparing the exact sum with the values returned by `plain_add` and `sort_add`. Do enough experiments until you see a pattern emerging. In particular, what is the influence of the length of the list and the value of `Digits` on the errors of `plain_add` and `sort_add`?

Assignment Two

1. Explain the difference between the two summation algorithms. Why is it better to first sort the numbers before adding them?
2. Maple sorts in increasing order. Illustrate with an example why for the summing problem this is numerically better than sorting in decreasing order.

2. Summing special sequences

We now consider a very special sequence of numbers. Our special sequence is determined by three parameters: (x,n,d) . x is just a random number, n is the number of elements in the list, and d is the precision used to calculate the sequence, i.e.: d is the length of the numbers in the sequence. Here is the Maple procedure:

```
[> very_special := (x,n,d) -> [seq(evalf(10^d*sin(x+k*(1.0+10^(-d+1))*Pi),d),k=1..n)]:
[> Digits := 10:          # restore value of Digits to default
[> x0 := stats[random,uniform[0,1]](1):
```

For example, to generate 6 numbers using x_0 with 50 decimal places, we type:

```
[> special_data := very_special(x0,6,50):
```

Now we calculate the sums using 30 digits as working precision:

```
[> Digits := 30;
[> plain_add(special_data);
                                0.
[> sort_add(special_data);
                                21
                                - .5 10
```

```
[> evalf(exact_add(special_data),60);
```

82.

Here we often see huge differences in the answers.

Assignment Three

For which values of “ d ” (precision used to calculate the sequence) and “Digits” (precision used to calculate the sums) do the numerical sums start to agree with the exact sum? Make a two-dimensional table (values for d in rows, values for Digits in columns) with content the difference between the exact sum and the sum calculated by `sort_add`, for appropriate ranges of d and Digits to illustrate the point.

Assignment Four

Explain why this special sequence is so difficult to compute by numerical methods.

3. Deadline is Friday 12 September 2003 at 1PM

Bring your solution to the project to class. It should contain the following:

1. The tables with numerical values of the experiments you have done.
2. Answers to the questions in the assignments. Please write complete grammatically correct sentences.
3. A print out of the Maple worksheet with the set up of your experiments. Suppress long output with colons.

Do not e-mail me Maple worksheets. The solution to the project is essentially a report on paper.

If you have questions or difficulties with the assignments, feel free to come to my office for help.