CSE/MATH 455, Arnold Lab 1 September 12, 1997

Errors in finite precision arithmetic

Do as much as you can during class, and complete after class if needed. Then write up your results and explanations clearly and carefully on separate sheets to hand in. Don't hand in this lab sheet. *Make your explanations as clear and precise as you can.*
Due date: September 19.

1. (a) Compile and run this program to read a number n and compute the sum $\sum_{j=1}^{n} \frac{1}{j^2}$. (We use Fortran here rather than Matlab because we want single precision, and Matlab only has double. Use C if you prefer.)

```
print*,'enter n'
read*,n
sum = 0.
do j = 1,n
  sum = sum + 1.0/j**2
enddo
print*,sum
end
```

Using the program complete the following table.

n	10	100	1,000	10,000	50,000
sum					

What value does this suggest for the infinite series $\sum_{j=1}^{\infty} \frac{1}{j^2}$?

(Extra credit: Try entering $n = 100,000$ and explain the result.)

(b) Change the `do` statement to `do j=n,1,-1`, so it goes through the numbers from largest to smallest. Give the results of this program:

n	10	100	1,000	10,000	50,000
sum					

What is the apparent value of $\sum_{j=1}^{\infty} \frac{1}{j^2}$ this time?

(c) The infinite series is in fact known to be exactly equal to $\pi^2/6$. Which program for computing the sum was more accurate? Discuss why this is the case (you may want to do part (d) first.)

(d) It is helpful to think of an extreme case. What would be the result of computing the following sum in single precision floating point arithmetic in the order given, and in reverse order?

$$100,000,000. + \underbrace{1. + 1. + \cdots + 1.}_{\text{1,000 times}}$$

(You shouldn't need the computer for this, although you can use it to check.) Explain.

2. Matlab has a function `chop(x,n)` which rounds (not chops!) the number x to n decimal places. Familiarize yourself with this function by computing, e.g., `chop(pi,2)`, `chop(pi,3)`, ... Give the command `format long` to see 15 decimal places.

Use the following Matlab statement to generate a list of 1,000 random floating point numbers between 0 and 20:

```
x = 20*rand(1,1000);
```

Chop all the numbers in the list to four decimal places:

```
xapprox = chop(x,4);
```

Compute the relative error: `relerr=(x-xapprox)./x;`. Theoretically, what is the largest magnitude the relative error can be? Check this in two ways: `max(abs(relerr))` and `plot(relerr)`.

Now examine how the relative error introduced by `chop` gets propagated by some functions. Take the square root of all the numbers in the list `x`, and see how well the square root of `xapprox` approximates them:

```
s=sqrt(x); sapprox=sqrt(xapprox); srelerr=(s-sapprox)./s;
```

Plot the relative error in the square root. Put the new plot on top of the previous plot by typing `hold on` before plotting. You'll need to plot in a different color to see the new plot. Use, e.g., `plot(srelerr,'r')` will plot in red. (Do `help plot` for more colors.) How do the relative errors in the x and $\sqrt{x}$ compare?

Do the same thing for at least one other function, e.g., x^2, x^3, $\exp(x)$, ... Try to explain what you see.