Symbolic vs Numerical Computing

Objectives

- Define symbolic computing and numerical computing
- Explore the difference via an example

An example via iteration

- Consider iterating a formula to compute $\sqrt{3}$
- Suppose we have a square with area 3; then the length of each side is $\sqrt{3}$
- Drawing a square the correct size is not easy. Consider a rectangle with sides with lengths *x* and *3/x*; it's area is 3.
- In order to obtain a square with the right area, we could compute a new x that averages the side lengths of the rectangle, and repeatedly do this to try to obtain our answer:

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{3}{x_n}\right), \quad n = 1, 2, 3, \ldots.$$

- One can prove that $\lim_{n\to\infty} x_n = \sqrt{3},$ as desired provided $x_1 > 0.$

Symbolic version

- Trying the iteration in Mathematica…
- First two iterates:

In[1]:= $x_1 = 5/3;$

In[2]:= $n = 1;\ x_{n+1} = \frac{1}{2}\left(x_n + \frac{3}{x_n}\right)$

Out[2]= $\frac{26}{15}$

In[3]:= $n = 2;\ x_{n+1} = \frac{1}{2}\left(x_n + \frac{3}{x_n}\right)$

Out[3]= $\frac{1351}{780}$

- The fraction is exact, but is getting more digits in the numerator and denominator

Symbolic version

- Now the next three iterates:

In[4]:= $n = 3;\ x_{n+1} = \frac{1}{2}\left(x_n + \frac{3}{x_n}\right)$

Out[4]= $\frac{3\,650\,401}{2\,107\,560}$

In[5]:= $n = 4;\ x_{n+1} = \frac{1}{2}\left(x_n + \frac{3}{x_n}\right)$

Out[5]= $\frac{26\,650\,854\,921\,601}{15\,386\,878\,263\,120}$

In[6]:= $n = 5;\ x_{n+1} = \frac{1}{2}\left(x_n + \frac{3}{x_n}\right)$

Out[6]= $\frac{1\,420\,536\,136\,104\,448\,487\,712\,806\,401}{820\,146\,920\,573\,494\,197\,299\,310\,240}$

- The fraction is now getting many digits , and gets much worse very quickly: it roughly doubles with each step

Symbolic version

- How is the error doing? The error is the difference between the iterate and the exact answer

Table 1: Errors in the *Mathematica* iterations of the square root algorithm.

n	$x_n - \sqrt{3}$	n	$x_n - \sqrt{3}$
1	-0.06538	4	6.499×10^{-14}
2	1.283×10^{-3}	5	1.219×10^{-27}
3	4.745×10^{-7}	6	4.292×10^{-55}

- It is converging very well: note how the *exponent* is doubling each time!

Numerical version

- Now try the iteration in Matlab
- Start with same guess
- Each iterate has 16 digits, so the approximation doesn’t require more memory each time, but it is not exact.
- Still, the answer converges to the correct answer within the ability of Matlab to represent the answer (not exact)

```
>> x = 5/3;
>> n = 1;  x(n+1) = ( x(n)+3/x(n) ) / 2;
>> n = 2;  x(n+1) = ( x(n)+3/x(n) ) / 2;
>> n = 3;  x(n+1) = ( x(n)+3/x(n) ) / 2;
>> n = 4;  x(n+1) = ( x(n)+3/x(n) ) / 2;
>> n = 5;  x(n+1) = ( x(n)+3/x(n) ) / 2;
>> format long,  x'

ans =

   1.666666666666667
   1.733333333333333
   1.732051282051282
   1.732050807568942
   1.732050807568877
   1.732050807568877
```

Numerical version

- The error for all of the iterates is shown at right
- The answer is too small to show with 16 digits and without scientific notation in just a few iterations
- So, for a smaller amount of fixed memory, we can rapidly get good answers this way as well.

```
>> x' - 1.7320508075688772935274 5

ans =

  -0.065384140902210
   0.001282525764456
   0.000000474482405
   0.000000000000065
                   0
                   0
```

Consequences of numerical computing

- Sometimes (not commonly) rules of arithmetic don't hold
- As a simple example, consider these two calculations which round the second decimal place. Only the grouping and thus order of operations changes.

$$(1.11 + 0.00411) + 0.00411 = 1.11\cancel{411} + 0.00411 = 1.11\cancel{411},$$

$$1.11 + (0.00411 + 0.00411) = 1.11 + 0.00822 = 1.11\cancel{822},$$

- The first results rounds to 1.11, while the second rounds to 1.12.
- This can be exacerbated in certain problems or with certain (undesirable) algorithms.
- We will study ways to avoid these problems, among other things.