Numerical Methods

Haefner Ch 6

- Number storage in computers
- Computation errors
  - fault of computer
  - fault of numerical methods
- Numerical methods for solving population DEs

Model Solutions (mathematical models)

A) Analytical

Symbolic manipulation of equations

$y = x^2$, want to know x

$x = \sqrt{y}$

B) Numerical

Equations not manipulated symbolically
Solution provided as numerical data

$y = x^2$ What is x if $y = 5$?
Repeated guessing, interval halving

Storage of numbers in computers

Integers – exact values maintained

Type	Bits	Range
char	8	±128
short	16	±32768
long	32	±2E9

Floating point numbers

Type	Bits	Digits	Range
Float	32	6	$\pm 10^{\pm 38}$
Double	64	15	$\pm 10^{\pm 308}$

Stores very large and very small numbers – What's the problem ??

The problem - All numbers cannot be represented

Why do we care ?

Remember, the number line has infinite points between 0 and 1.

$$y = a + b$$

$$\begin{array}{r} 1.500 \\ +2.000 \\ \hline 3.500 \end{array}$$

Simplified example in base 10 :

Can store 4 digits : ___ ___ ___ ___

Allows for 0 to 9999
Want to store bigger numbers ?

Divide up storage space : ___ ___ ___|___

mantissa 0-999 | exponent 0-9

number = mantissa $*$ 10^{exponent}

exponent	range	resolution
0	0-999	1
1	0-9990	10
2	0-99900	100
9	0 - 999 billion	1 billion

9985 is not possible !!

Problem adding large and small numbers :

$$\begin{array}{rcl} 1\,0\,0\,1 & = & 1\,0\,0\,0 \\ +\,0\,0\,1\,0 & = & +\quad\;\; 1 \\ \hline 1\,0\,1\,1 & \neq & 1\,0\,0\,1 \end{array}$$

= 1010

What really happens :

1000 + 0010 = 1000

too small of a change !!

More complicated with floats, but same idea…

```
#include <stdheaders.h>

int main( void )
{
        int i;
        float y, a, b;

        a = 3e10;
        b = 1.0;

        y = a + b;

        printf( "\n\n a = %14.2f\n b = %5.2f\n y = %14.2f\n", a, b, y );

        return 0;
}
```

```
c:\lcc\bin\rundos.exe

 a =  30000001024.00
 b =      1.00
 y =  30000001024.00

"c:\lcc\projects\temp\lcc\temp.exe"
Return code 0
Execution time 0.016 seconds
Press any key to continue...
```

Numerical solution of DEs

Exponential population model :

$$\frac{dN}{dt} = kN$$

What is N for future t ?

Simplest numerical solution – Euler's method

$$\frac{\Delta N}{\Delta t} \approx kN$$

$$\Delta N \approx kN\Delta t$$

$$N_{t+1} = N_t + \Delta N$$

```
#include <stdheaders.h>   // include header files with input/output

int main( void ) // main function is program entry point

{
        int i;           //These three lines declare the variables used
        float k, dN, dt;
        float N[101], t[101];     //Declare space for 101 rows for N and t

        k = 0.4; //These four lines set some starting values
        t[0]= 0;
        N[0] = 2;
        dt = 1.0;        //This one is for the time step size

        printf( "Exponential Growth Model\n\n" );
        for( i=0; i<10; i++ )     //Loop calculates pop. sizes
        {
                dN = k * N[i] * dt;       //Calc pop growth
                N[i+1] = N[i] + dN;       //New size is old + growth
                t[i+1] = t[i] + dt;       //New time is old + time step
        }
        for( i=0; i<=10; i++ )    //Loop prints results
        {
                printf( "At time %4.1f, N = %7.4f\n", t[i], N[i] );
        }

        return 0;
}
```

Logistic Model and DE solutions

$$N_{t+1} = N_t + rN_t\left(1 - \frac{N_t}{K}\right)$$

$$\frac{dN}{dt} = rN\left(1 - \frac{N}{K}\right)$$

Can modify exponential program by substituting the RHS parts.

$$\frac{dN}{dt} = kN \longrightarrow \texttt{dN = k * N[i] * dt;}$$