Lecture 2

- Fortran
- Numbers

Outline

Fortran

- Fortran advantages and disadvantages
- Compiling a fortran program
- What we won't be covering

Fortran Advantages

- Compiled languages are quick
- Free (good) compilers available
- Type safe: protects you from some errors
- Good free and commercial libraries
- Lots of academics speak fortran
- Designed for maths: complex numbers, raising to a power

Fortran Disadvantages

- Unpopular outside academia (Java, C++)
- Longer programs - more scope for mistakes
- Dialect misery: Fortran 77, 90, 95, (2003, 2007). E.g. a lot programs allowing fortran extensions want F77.

Compiling Fortran

- Compiling turns an ASCII file into an executable binary file
- An intermediate stage of creating object files also possible
- Useful for large projects, only recreate changed object files then compile all object files into executable

What we won't be covering

- Interoperability with C, Matlab or any other program
- Except by I/O
- C(++) features, pointers, objects
- Parallel programming

Outline

Fortran

Error, accuracy and stability

Numbers

How numbers are represented in a computer

- Numerical disasters
- Numeric datatypes in Fortran and Matlab
- How a computer stores numbers
- Floating point arithmetic
- Disasters revisited

Numerical Disasters

- June 4, 1996. Ariane rocket explodes after going off course on liftoff.
- 1991. Patriot missile fails to intercept Iraqi scud missile during 1st Gulf war.

Both these disasters were caused by a failure to appreciate that a computer does not allocate infinite space to storing numbers.

Numerical Datatypes

- Integers: ...,–4,–3,–2,–1,0,1,2,3,4. Beware: integer arithmetic rounds down 1/2=0!!!. Convert integers to reals before dividing
- Real numbers.
- Complex numbers.

Matlab does not have an integer data type. Integer arithmetic is exact except for overflow.

Floating point numbers

Computers approximate the real numbers $\mathcal{R}$ by the floating point numbers $\mathcal{F}$.
$\mathcal{R}$: $1/7 = 0.\overline{142857}$
$\mathcal{F}$: $1/7 = 1.42857142857143e-01$ (Matlab, `format long e`)
Inside the computer a floating point number x is stored in the form

$$x = (-1)^s \times (0.a_1 a_2 \ldots a_t) \times \beta^e \tag{1}$$

$s = 0, 1$. $\beta \geq 2$ is basis. $(0.a_1 a_2 \ldots a_t)$ a set of t digits $0 \geq a_i \geq \beta - 1$, is the mantissa. $L < e < U$ is the exponent which adopts a finite range between $L < 0$ and $U > 0$.

$$x = (-1)^s \times (0.a_1 a_2 \dots a_t) \times \beta^e \quad L < e < U \tag{2}$$

The set of floating point numbers is determined by (matlab values) $\beta = 2$, $t = 53$, $L = -1021$ and $U = 1024$.
53 binary digits correspond to about 15 decimal digits, all of which are displayed by format long.
The error that is introduced by approximating x by $fl(x)$ is

$$\frac{|x - fl(x)|}{|x|} \leq \frac{1}{2}\epsilon_M \quad \epsilon_M = \beta^{1-t} \tag{3}$$

$\epsilon_M = 2.22044604925031e-16$ is given by the matlab `eps` command.

The largest possible number `realmax`$=1.79769313486232e+308$
The smallest possible number
`realmin`$= 2.22507385850720e-308$
In matlab a number greater than realmax is labelled `Inf`. In a general program the results are undefined. E.g. Fortran, some compilers produce Inf. Our compiler crashes the program. Worse still, some compilers will produce a random number *without telling you anything is wrong!*

Properties of floating point numbers

- Commutativity: $fl(x+y) = fl(y+x), fl(yx) = fl(xy)$
- No associativity $a+(b+c) \neq (a+b)+c$
- Zero is not unique

```
octave:8> x=1e-15; ((1+x)-1)/x
ans =  1.1102
```

Error is quite large.

Roundoff error

- Naïve estimate: N operations, floating point error $\sqrt{N}$ eps. (Random walk.)
- Often errors all in same direction: N eps
- Sometimes errors much bigger than this: usually due to nearly cancelling subtraction

E.g. quadratic formula

$$\frac{-b+\sqrt{b^2-4ac}}{2a} \tag{4}$$

$b^2 \gg 4ac$

Truncation Error

- Software rather than hardware limitation
- Due to approximating continuous function or infinite series by finite set of points.

Stability

Errors (roundoff, experimental) introduced at start of algorithm magnified at each iteration.
E.g. powers of golden ratio

$$\phi = \frac{\sqrt{5}-1}{2} \approx 0.618 \tag{5}$$

$$\phi^n = \phi^{n-1} - \phi^{n-2} \tag{6}$$

- $-\frac{1}{2}\left(\sqrt{5}+1\right) > 1$ is also a solution to recurrence relation
- Roundoff error introduces a tiny bit of it
- Grows exponentially with repeated iterations

```
n=50
phi1 =  1;
phi=(sqrt(5)-1)/2;
phi2 =phi;
for i=2:n
  phi3=phi1-phi2;
  fprintf("%8d %18.6e  %18.6e\n",i,phi3,phi^i)
  phi1=phi2; phi2=phi3;
end
```

```
      45      6.204330e-08        3.940544e-10
      46     -9.950704e-08        2.435390e-10
      47      1.615503e-07        1.505154e-10
      48     -2.610574e-07        9.302363e-11
      49      4.226077e-07        5.749176e-11
      50     -6.836651e-07        3.553186e-11
```

Ariane Rocket

On board computer tried to convert a 64 bit real to a 16 bit integer and failed because the number was too large to have an integer representation.
There was a backup computer duplicating its functions but that failed for the same reason.

Patriot Missile

Clock time (24 bit real) incremented at 0.1 second intervals, but 0.1 does not have exact floating point representation. After 100 hours error 0.3 s. A missile travels a long way in 0.3 s

Worksheet 2. Hello Fortran.