Programing Techniques for Scientific Simulations: Report 1

Marion Baumgartner

October 2, 2012

1 Part A

1.1 Endianness

The endianness basically tells us how the bytes are ordered. We say that a machine has little endianness if the least significant byte is first. If the most significant byte is first then the machine has big endianness. If we take for example a 32 bit integer which is stored in 4 bytes we get
Byte 3; Byte 2; Byte 1; Byte 0

In a little endian we have the following arrangement:
BaseAddress+0 Byte0
BaseAddress+1 Byte1
BaseAddress+2 Byte2
BaseAddress+3 Byte3.

For a big endian we get
BaseAddress+0 Byte3
BaseAddress+1 Byte2
BaseAddress+2 Byte1
BaseAddress+3 Byte0.

In order to check weather a machine has little or big endianness we can simply check how it sores an n-bit integer (this takes n/8 bytes)

1.2 Machine Epsilon

The machine ϵ is the smallest number such that $1 + \epsilon \neq 1$.

What the program does is it take an initial $\epsilon = 1$ and then check if $1 + \epsilon \neq 1$ if this is true then we decrease *epsilon* by $\frac{1}{2}$ and check again. This is repeated until we find an ϵ which does not fulfill the criteria. At this point we know that the ϵ we had one step before is what we are looking for. This can be done for float, double and long double. The main idea is always the same, just the data types change.

2 Part B

2.1 Simpson's rule

In this part the Simpson's rule was used to integrate $\int_0^\pi \sin(x)dx$

The basic formula used is

$$Q(f) = \frac{h}{3} \cdot \left(\frac{1}{2} f(x_0) + \sum_{k=1}^{N-1} f(x_k) + 2 \sum_{k=1}^{N} f\left(\frac{x_{k-1} + x_k}{2} \right) + \frac{1}{2} f(x_N) \right) \tag{1}$$

$$h = \frac{b-a}{N} \qquad\qquad x_k = a + k \cdot h$$

Where $[a, b]$ is the interval to be integrated over and N determines the number of bins (N has to be even). Further we have that

$$\int_a^b f(x)dx = Q(f). \tag{2}$$

The program calculate equation (1) for a given, N, a, b and the function $f(x) = \sin(x)$.