ASTR4004/ASTR8004
Astronomical Computing
Lecture 10

Christoph Federrath

30 August 2016

Monte Carlo error propagation, C/C++ startup

1 Monte Carlo error propagation

1. Uncertainty analysis of measurements and derived quantities In astrophysics is extremely important, and published measurement results should always have a proper error analysis associated with them. Here we learn how to use IDL to do Monto Carlo (MC) error propagation, which goes beyond standard analytic error propagation and can handle non-Gaussian distributions for propagating uncertainties.

2. Suppose you have measured two Gaussian random variables,

$$a = 10 \pm 1 \qquad \text{and} \qquad b = 1.0 \pm 0.1, \tag{1}$$

   and you want to calculate the derived quantity

$$c = \frac{a^2}{b^2}. \tag{2}$$

3. First, calculate the uncertainty of c using standard analytic methods of error propagation.

4. Now write a program that does the MC error propagation. First make Gaussian random numbers and define a and b based on these Gaussian random distributions. Then define c based on a and b.

5. Now plot the PDFs of a, b, and c. What is special about the PDF of c? Try also a log-y axis version.

6. Compute the mean and standard deviation of c based on the PDF of c and compare to the analytic estimate. Make sure to implement bin-centered binning for the numerical integration of the PDF, in order to recover the mean and standard deviation more accurately than from the staggered bins.

7. Get the mode (most probable value) of c and the 16th and 84th percentile values.

8. Try changing the sample size of the Gaussians (`NRAND`) and the number of bins used for the PDFs, in order to study numerical convergence of the results.

2 Introduction to C/C++

1. IDL is very good for exploring datasets in an interactive and scripted way. Many operations can be coded up in one line in IDL (e.g., the computation of the sum or the mean of an array). However, the downside of this is that these operations can use up a lot of memory or that they are relatively slow. Most importantly, extremely big datasets cannot be handled in IDL, because IDL does not parallelise beyond a node. In order to write programs that can be parallelised across nodes and even across different architectures, we have to move to a different programming language. The most popular one is C/C++.

2. Lets check all the requirements to write and compile C/C++ programs. You can use `misfit`, which has all the required compilers (`gcc` and `g++`) and libraries already installed.

3. We start by writing a simple C program that prints something to the screen and reads a number from the user.

4. Then we write a C++ program and use of a `Makefile` to automate the compiling and building of the code.

5. Now lets make a loop that sums up a ton of numbers and times the computation.

6. Finally, we use the `openmp` library to parallelise the loop and see by how much the execution time can be reduced with this method of parallelisation.