PH502 Practical 2

ICHEC

January 12, 2018

Overview

The second PH502 practical will use many techniques previously discussed including: loops, conditionals, input/output, Makefile and version control.

Exercises

Please carry out the following exercises.

1. It is always good practice to use version control while working with source code. Throughout this practical you will use the distributed version control system called git (`http://git-scm.com/`).

   (a) Set up your git environment:

   ```
   module load dev git/intel/2.2.1
   git config --global user.name 'Your Name Comes Here'
   git config --global user.email you@yourdomain.example.com
   ```

   (b) Create a folder called 'practical2.git'. This will act as a central repository. Enter into this folder and initialise a bare git repository for your code.

   ```
   mkdir practical2.git ; cd practical2.git; git --bare init
   ```

   (c) Now clone this repository and add a README file:

   ```
   cd ../ ; git clone practical2.git practical2.clone;
   cd practical2.clone; touch README
   ```

   The README file is a common file found in software packages that gives users some information about the package.

   (d) Tell git that the README exists:

   ```
   git add README
   ```

   (e) The snapshot of the code is stored in a temporary staging area ("see 'git status'") which git calls the "index". You need to permanently store the contents of the index in the repo with git commit commands:

   ```
   git commit -m 'Initial version of the README file' README
   ```

   (f) Now to add this change to the central repository we will need to carry out a 'push':

   ```
   git push origin master
   ```

   (g) Add two subfolders called pi and mm for two programs which you will write later.

   (h) Add, commit, push these subfolders. You may need to add a README files to each because git doesn't like empty folders.

2. Write a program in the pi subfolder of practical2.clone to calculate π using the following equation:

$$\pi = \int_0^1 \frac{4.0}{1+x^2} dx$$

which can be approximated by the following sum:

$$\pi = \frac{4}{\mathrm{N}} \sum_{k=1}^{\mathrm{N}} \frac{1}{1+h^2}$$

where $h = (k - 0.5)/\mathrm{N}$ and the only input data required is N.

After successfully completing each of the following stages please commit your code changes with descriptive messages. When finished issue the 'git log' command and review commits.

   (a) Read in a value for N from the command line.
   (b) Ensure that N is positive.
   (c) Loop from 1 to N and compute function above.
   (d) Print value of π and compare value to the following: $(4.0 * atan(1.0))$.
   (e) Push your commits to the central repo for safe keeping.

3. Write a program in the mm subfolder of practical2.clone to multiply two matrices, $\mathrm{C}_{n\times q} = \mathrm{A}_{n\times p}\mathrm{B}_{p\times q}$. See Fig. 1 for an illustration of the problem. Again after successfully completing each step please commit your code changes with descriptive messages.

   (a) Define $n = 5$, $p = 3$, and $q = 4$.
   (b) Declare three arrays A, B, and C of type double or real.
   (c) Initialise the C to zero and
      i. $\mathrm{A}_{ij} = i + j$,
      ii. and $\mathrm{B}_{ij} = i - j$.
   (d) Use the schematic below to determine the elements of C.
   (e) Print out the three arrays (with one row of the matrix per line) to the screen and ensure that you are generating the correct result.
   (f) Push your commits to the central repo for safe keeping.

4. In this exercise you will use Makefiles to build your programs created above.

   (a) Please create a Makefile to compile the π program by adding a single target to build the program
   (b) Rewrite the matrix-matrix multiplication program so that you have two files, main and func where the latter holds a function or subroutine where the multiplication is carried out.
   (c) Create a Makefile that compiles both the files and links the two together.
   (d) Add two additional targets:

i. 'clean' that removes all object files.
ii. 'opt' that uses additional compiler optimisation flags.

5. Create C and FORTRAN main programs and C and FORTRAN functions in separate files.

   (a) Create a function in C and FORTRAN that calculates the inner product (dot product) between to length 5 vectors v and u.

$$\text{dot prod} = \sum_{i=1}^{5} v_i u_i.$$

   (b) Create a program in C and FORTRAN that uses the above function to calcuate the inner product between these two vectors

$$u = \quad v = \quad (1,2,3,4,5)$$

   (c) Use a makefile to generate four programs which take all combinations of FORTRAN and C mains and functions. 'Load the gcc module and use "gcc" and "gfortran" as the compilers.

   (d) Test them to see if they give the same result.

   (e) Add and commit the sources to the git repo.

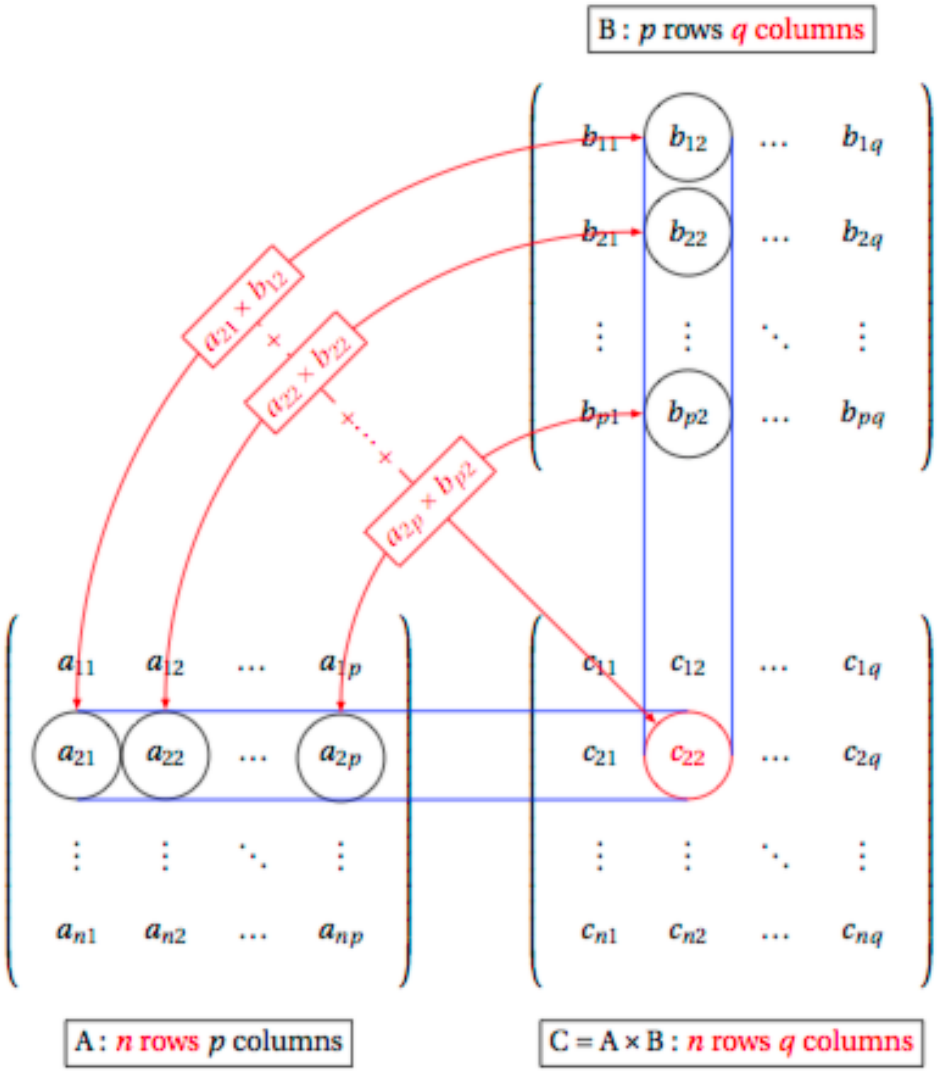

Figure 1. Matrix-Matrix Multiplication Schematic (Taken from: `http://www.texample.net/tikz/examples/matrix-multiplication/`).