|epcc|

MSc in High Performance Computing
MSc in High Performance Computing with Data Science
Programming Skills Coursework 2 - Development

Adrian Jackson, Mark Bull, David Henty, Alan Simpson, Mike Jackson

September 18, 2015

1 Introduction

In this coursework you will implement a sequential version of a two-dimensional predator-prey model with spatial diffusion using Fortran, C, C++, Java or Python. The aim is to produce a "best practice" scientific code within a full development framework (i.e. revision control, build tools, unit test framework etc).

The coursework is a group exercise where you will collaborate with your fellow students in small groups to implement the model. The groups are randomly assigned by the course organiser, based on the programming languages you are familiar with.

Your mark is based upon your group's source code. You also submit a, non-assessed, contribution questionnaire.

2 Modelling pumas and hares in a landscape

You are going to model the interaction of pumas (predator) and hares (prey) in a two-dimensional landscape. Pumas eat hares. Without hares, pumas decline and die out through lack of food. Without pumas, hares breed and increase in numbers forever. Both animals tend to spread out (or diffuse) to fill parts of the landscape where there are fewer of their own kind. Their landscape contains areas of water where neither pumas nor hares can live.

Partial differential equations can be used to model the behaviour of pumas and hares within a landscape:

$$\frac{\partial H}{\partial t} = rH - aHP + k\left(\frac{\partial^2 H}{\partial^2 x} + \frac{\partial^2 H}{\partial^2 y}\right)$$

$$\frac{\partial P}{\partial t} = bHP - mP + l\left(\frac{\partial^2 P}{\partial^2 x} + \frac{\partial^2 P}{\partial^2 y}\right)$$

where:

- H is the density of hares (prey).
- P is the density of pumas (predators).

- r is the birth rate of hares.
- a is the predation rate at which pumas eat hares.
- b is the birth rate of pumas per one hare eaten.
- m is the puma mortality rate.
- k is the diffusion rates for hares.
- l is the diffusion rates for pumas.

Do not worry if you don't understand these equations, as discrete versions that can be used in your program will be introduced shortly. What the equations say is that:

- The change in hares over time is affected by the birth rate of hares, the rate at which hares are killed by pumas and the diffusion of hares through the landscape.
- The change in pumas over time is affected by the birth rate of pumas, which also depends on the hares available as food, the natural death rate of pumas, and the diffusion of pumas through the landscape.

2.1 Numerical approximation

The mathematical model above treats time and space as continuous quantities but continuous quantities cannot be handled directly by a computer. We need to use a discrete approximation. To do this, we can divide space and time up into discrete units. The landscape can be modelled as a rectangular grid, of dimension $N_x \times N_y$, where N_x is the number of columns and N_y is the number of rows, and where each grid square can be either land or water. Within each grid square there will be a number of hares, H_{xy}, and pumas, P_{xy}, living in it, where H_{xy} and P_{xy} are >= 0.

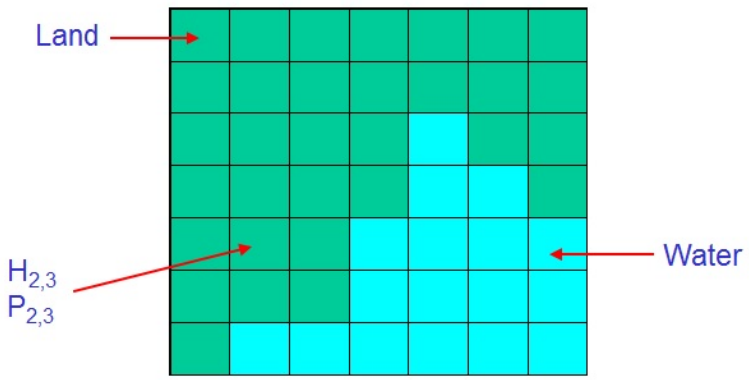

Figure 1: 2-dimensional landscape grid. Squares are indexed by column then by row from the bottom-left.

Time can be modelled by successive, discrete, time-steps, of size Δt. The numbers of hares and pumas in the landscape at next time step depend on the numbers present in the current time-step. So, the new number of hares, H, at time $t + \Delta t$, in a specific grid square, is given by:

$$H_{ij}^{\text{new}} = H_{ij}^{\text{old}} + \Delta t \left(r H_{ij}^{\text{old}} - a H_{ij}^{\text{old}} P_{ij}^{\text{old}} + k \left((H_{i-1j}^{\text{old}} + H_{i+1j}^{\text{old}} + H_{ij-1}^{\text{old}} + H_{ij+1}^{\text{old}}) - N_{ij} H_{ij}^{\text{old}} \right) \right)$$

The number of hares in a grid square at the next time step equals the number of hares there at present adjusted by the change in population of hares over the timestep. The population change is a combination of the number of hares born in that square (where r is the hare birth rate), the number of hares

killed in that square (which depends on number of pumas in the square, where a is the puma predation rate) and the number of hares that move into or out of that square from the neighbours (where k is the diffusion/migration rate).

N_{ij} is the number of "dry" (i.e. non-water) grid squares out of the four squares neighbouring the square i, j (to the north, south, east and west). From this approximation, we assume that hares cannot move diagonally nor can they swim.

The new number of hares, P, at time $t + \Delta t$, in a specific grid square, is given by:

$$P_{ij}^{\text{new}} = P_{ij}^{\text{old}} + \Delta t \left(bH_{ij}^{\text{old}} P_{ij}^{\text{old}} - mP_{ij}^{\text{old}} + l \left((P_{i-1j}^{\text{old}} + P_{i+1j}^{\text{old}} + P_{ij-1}^{\text{old}} + P_{ij+1}^{\text{old}}) - N_{ij} P_{ij}^{\text{old}} \right) \right)$$

The number of pumas in a grid square at the next time step equals the number of pumas there at present adjusted by the change in population of pumas over the timestep. The population change is a combination of the number of pumas that die in that square (where m is the puma death rate), the number of pumas born in that square (which depends on number of hares in the square where b is the puma birth rate) and the number of pumas that move into or out of that square from the neighbours (where l is the diffusion/migration rate). Pumas, like hares, cannot move diagonally, nor can they swim.

We can assume that the distance between grid squares is 1 i.e. $\Delta x = \Delta y = 1$, as scaling can be absorbed into the diffusion terms, k and l.

2.2 More on the birth and death of hares and pumas

Let's look at the birth of hares:

$$H_{ij}^{\text{old}} + \Delta t \left(rH_{ij}^{\text{old}} ... \right)$$

As an example, let's take the basic unit of time as 1 month, and let's assume that every pair of hares breeds 3 times a year (once every 4 months) and each produces a litter of 5 baby rabbits (called leverets). So, in 4 months, 2 hares produce 5 more hares, so each hare produces 0.625 hares a month, so the hare birth rate, r is 0.625. To check that this is correct, let's substitute into our equation: $H_{ij}^{\text{new}} = 2 + 4(0.625 \times 2)$ which equals 7 (the 2 parents and the 5 offspring).

As we could have thousands of hares breeding continuously, we can use a much smaller value of Δt. A value for r can depend upon the units chosen e.g. months, days etc. Likewise, it can improve the scalability of our model if, instead of recording the hare, and puma, populations as integers, we record them as real numbers in terms of densities per grid square e.g. 3.53 hares per hectare.

As for the death of hares:

$$H_{ij}^{\text{old}} + \Delta t \left(rH_{ij}^{\text{old}} - aH_{ij}^{\text{old}} P_{ij}^{\text{old}} ... \right)$$

Hares only die through being eaten by pumas, not naturally, so more pumas means more hares are killed. More hares also means more are killed as they are easier to find.

Looking at the birth of pumas:

$$P_{ij}^{\text{old}} + \Delta t \left(bH_{ij}^{\text{old}} P_{ij}^{\text{old}} ... \right)$$

Unlike hares, the birth rate of pumas (b) depends not only on the number of their own kind but also on their available food, the hares.

As for the death of pumas:

$$P_{ij}^{\text{old}} + \Delta t \left(bH_{ij}^{\text{old}} P_{ij}^{\text{old}} - mP_{ij}^{\text{old}} ... \right)$$

Pumas differ slightly in that, unlike hares, they have no predators, so die at a natural rate (m).

2.3 More on movement of hares and pumas across the landscape

We could just model a single population over the whole landscape, but this is not very realistic. We want a finer-grained model which allows for different densities of pumas and hares at different points in the landscape. Our model will be more realistic if we model how animals move around the landscape. For hares, we have a diffusion term, k:

$$k\left(\frac{\partial^2 H}{\partial^2 x} + \frac{\partial^2 H}{\partial^2 y}\right)$$

In the discrete form of our equation, this is represented as a "random walk":

$$k\left((H^{\text{old}}_{i-1j} + H^{\text{old}}_{i+1j} + H^{\text{old}}_{ij-1} + H^{\text{old}}_{ij+1}) - N_{ij}H^{\text{old}}_{ij}\right)$$

The number of hares in surrounding squares is summed. From this is subtracted the number of hares in the current grid square multiplied by the number of surrounding land-only grid squares (N_{ij}). The total value is then multiplied by a hare diffusion term, k.

For pumas a similar calculation is used, with a puma diffusion term, l.

3 Starting your coursework

Groups of between 3-4 students are randomly assigned by the course organiser, based on the programming languages you are familiar with.

You should get together in your group and create your design and list of tasks. The tasks are up to your group and do not have to be single-person tasks, you are free to work together as you think best. You should ensure that the overall amount of work each member of your group does is equally divided.

Here are some examples of tasks that could be suggested (although not all of these may be appropriate for your group's design and task breakdown).

- Set up and maintain a source code repository (e.g. Subversion or Git).
- Create the build environment.
- Define tests.
- Implement the computational kernel.
- Implement the I/O part of the code.
- Implement the test framework.
- Write documentation.

Note that you are not required to use this list or any of the items on it for your tasks. It is provided as an example of the sorts of tasks you could consider.

Once the tasks have been defined you should assign them amongst the group members to ensure that each person does (roughly) equal amounts of work. As development proceeds it may be necessary to change the tasks or re-assign work to ensure that people have the correct amount of work to do. It will also be beneficial to assign tasks to people's skills wherever possible.

You should ensure that everyone has some coding/development work.

Any conflicts that cannot be resolved within the group will be mediated by the course organiser, on request.

4 Development

As it is your source code that is submitted and marked, you should use good programming practice such as comments, function descriptions and definitions, etc. You are strongly encouraged to put your code and tests under revision control, use a build tool (e.g. Make or ANT) and implement tests since you will be marked on these.

It is expected that all groups will get a working implementation.

4.1 Landscape

Your code must be able to handle landscapes up to 2000×2000 grid squares ($1 <= N_x <= 2000$ and $1 <= N_y <= 2000$).

Your code must be able to cope with arbitrary "water" grid squares, where hare and puma population values are always zero.

The hare and puma densities should be modelled with two-dimensional double precision floating point arrays. C programmers are strongly encouraged to use either statically declared arrays, or the `arralloc()` routine. This routine's (`.h` and `.c` files) are in the LEARN area for this coursework (see Appendix 9).[1]

The boundary conditions are $H = P = 0$ on the edges of the landscape. This can be implemented by adding a halo of grid squares around the edges of the landscape and making them all "water".

4.2 Inputs

Your code must be able to read in a bitmask ASCII file describing which points are land or water. The format of this file consists of one line giving N_x and N_y, column and row, dimensions, followed by a sequence of space separated ones and zeros (land=1, water=0). For example:

```
7 7
1 1 1 1 1 1 1
1 1 1 1 1 1 1
1 1 1 1 0 1 1
1 1 1 1 0 0 1
1 1 1 0 0 0 0
1 1 1 0 0 0 0
1 0 0 0 0 0 0
```

The LEARN area for this coursework (see Appendix 9) has examples. See Appendix 10 for how you can generate your own land/water mask files.

The user must be able to set all the parameters in the equations. These are as follows, with suggested default values in brackets:

- r is the birth rate of hares (0.08).
- a is the predation rate at which pumas eat hares. (0.04)
- b is the birth rate of pumas per one hare eaten (0.02).
- m is the puma mortality rate (0.06).
- k is the diffusion rates for hares (0.2).

[1] We provide `arralloc` to simplify the process of allocating memory in C as it generally causes a lot of problems for people and wastes a lot of time unnecessarily. `arralloc` wraps up all the complexities of `malloc` and provides a nice clean interface for creating arrays. However, if you are confident with your C and are happy to do your own memory allocation then feel free to do so.

- l is the diffusion rates for pumas (0.2).
- Δt, the size of the time step (0.4).

In addition, the user must also be able to set T, the number of time steps between outputs (see below).

You can use random initial densities of H and P, say in the range 0 to 5.0, or you could find out what happens when a large group of pumas are placed in a confined area, with a uniform distribution of hares.

Run the simulation from time $t = 0$ to $t = 500$.

4.3 Outputs

Your program must output:

- A "Plain PPM" file every T timesteps, where T is the number of time steps between outputs. "Plain PPM" is a simple uncompressed ASCII format (like PGM but colour). Run `man ppm` for details.
- Average values of H and P over the grid at regular intervals. These averages can be calculated across the grid as a whole, or just across land-only grid squares. How you output this (whether you put the values into a file or just print them to standard output) is up to you. The frequency of output is up to you, depending on how you wish to interpret "regular intervals" but the number of intervals must be greater than than 1.
- When the simulation completes, the total time that was taken to execute the model simulation. Again, how you output this is up to you.

4.4 Implementing the equations

Implement the discrete approximations of the equations, as they are defined above.

Using more accurate approximations of the time-derivatives and of the equations - using optimisations of the equations themselves derived numerical analysis techniques - should *not* be attempted. This is a programming skills exercise not a numerical analysis one, and an important aspect of programming skills is meeting the requirements of the customer, which, in this case is implementing the equations as defined above.

The implementation should be sequential, this is *not* a parallel programming exercise.

4.5 User interface

Your program must be runnable from the command-line in batch mode, and once started, your program should run the simulation and output the results without need for user interaction.

Whether you also support other interface approaches e.g. an interactive command-line interface or a GUI is up to you.

4.6 Testing

Testing is important for any code so you need to ensure that you can validate the results of the code you produce, especially if you make changes to fix bugs or optimisations.

Using a test framework can help to automate the running of your tests, and reporting of their results.

5 Source code

Please note that your code and tests must satisfy the following:

- It must be possible to build and run your code and tests on the Physics Computational Lab machines.
- It must be possible to build and run your code and tests from the command-line.
- If your code or tests requires additional packages or software to build or test that are not already available on the Physics Computational Lab machines then you should document what these packages or software are, where to get them, and how to set them up.

Non-working or non-compiling code will be marked down accordingly.

You must also provide some documentation, which should include:

- Information on the programming language, revision control, debuggers, build tools, and test tools you have used.
- How to build your code.
- How to run your code.
- How to run your tests.

6 Contribution questionnaire

Every member of the group should complete their own contribution questionnaire. This summarises what you as an individual contributed to the coursework and how you felt each other member of the group contributed. Only the markers will see the contribution questionnaires.

The LEARN area for this coursework (see Appendix 9) contains copies of the questionnaire:

- Microsoft Word, `ContributionQuestionnaire.doc`
- LaTeX, `ContributionQuestionnaire.tex`

7 Submission and marking

You are required to submit the following:

- Source code (this includes build scripts, test code etc) and documentation together in a `zip` or `tar.gz` file.
- Contribution questionnaire.

Submissions are done via the Turnitin submission tool from within LEARN, under Coursework Submission.

The filename of your source code submission and contribution questionnaire must both include your exam number (e.g `B123456`) which appears on your matriculation card. You should also include the course identifier, `PS` for Programming Skills and the identifiers for this coursework: `SC` for source code and `CQ` for the contribution questionnaire. For example `B123456-PS-SC.zip` or `B123456-PS-SC.tar.gz`, for your source code, and `B123456-PS-CQ.doc` or `B123456-PS-CQ.pdf` for your contribution questionnaire.

While every member of the group needs to submit the source code so that we can return marks to all students via Turnitin and LEARN, only one of these (per group) will be marked. The one to be marked will be picked from a member of the group at random.

7.1 Deadline

The deadline for submission your source code and questionnaires is **14:00 Friday 6th November (week 7) 2015** for all students.

Submission are allowed up to 5 calendar days after the deadline, with a 5% deduction per day or part-day late.

If you submit your coursework by the deadline then it will be marked, and you cannot submit an amended version later on, to be marked.

If you do not wish the submitted version to be marked after the deadline, and intend submitting an amended version after the deadline, then it is your responsibility to let the course organiser know, in advance of the deadline, that the submitted version should not be marked.

7.2 Marking

Marks will be awarded as follows:

- Source code
  - Correctness 20
  - Design 15
  - Presentation and readability 10
  - Use of tools and frameworks 5
  - **Sub-total 50**
- Documentation **Sub-total 10**
- **Total 60**

Your source code and documentation will be marked. Everyone in your group will get the same mark for these. The contribution questionnaires will be used to moderate the marks in cases where the contribution of effort is unbalanced across the group. However, if all group members view each other to have contributed at the same level then each individual will get the same mark.

To summarise,

$$FinalMark = ((SourceCode + Documentation) + / - Moderation)$$

8 Questions

Feel free to e-mail the course organiser if you have any questions. Questions of interest to the whole class will be anonymised and them, plus their answer, forwarded to the whole class.

There follows answers to common questions...

8.1 Can we use GitHub or BitBucket?

Yes. BitBucket is preferable as it allows you to create a private Git repository you can share amongst your team without allowing others to view it.

8.2 Sharing files and repositories across accounts

On the Physics Computational Lab machines, you can give each other read/write access to your files and directories (including any repository you create in your home directory) as follows.

Suppose your user name is `s12345`. First you need to ensure others can read your directory:

```
$ chmod go+rx /Home/s12345
```

If you have private files, that you don't want others to see, run the following on those files and directories:

```
$ chmod go-rx /home/s12345/your-private-file-or-directory
```

Do `man chmod` for more information.

If you have a directory you want a fellow student e.g. `s99999`, to be able to read/write, for example shared-directory, you can set up an access control list:

```
$ setfacl -R -m u:s99999:rwx shared-directory
```

This allows `s99999` to read, write or execute files in shared-directory and any sub-directories. You can then add other users the same way.

To disable this and remove `s99999`'s permissions you can do:

```
$ setfacl -R -x u:s99999: shared-directory
```

To see the current access control list you can do:

```
$ getfacl shared-directory
```

Do `man setfacl` and `man getfacl` for more information.

For repositories, users can then use them as they would a repository they have created in their home directory.

8.3 Can .PPM files and screen dumps be provided as evidence that we've met the requirements of the practical?

Yes, but these will be considered as supplementary evidence. The code itself is the primary evidence that you have met the requirements.

8.4 Can we use someone else's algorithm to calculate random numbers (so long as we put in a reference for it)?

Yes. Some languages provide built-in libraries for random number calculation, some do not. If your chosen language does not then feel free to use a third-party algorithm, library or package. Make sure you provide a reference for it and also that its licence allows you to use it (open source licencing is covered in Software Development next semester).

8.5 Other questions

- What should our code do if the population density becomes negative?
- How should the location of the landscape file be provided to the program?
- How robust should our program be in terms of valid or invalid user input values?

- How should we visualise the population of hares and pumas across the landscape at a specific point in time, using "Plain PPM" files?
- How do we get round the "Plain PPM" restriction that "No line should be longer than 70 characters"?
- Is it OK to use global variables?

This are all design decisions you need to make. There are many options available. Choosing an appropriate solution is one of the challenges of this coursework.

9 Appendix - coursework files on LEARN

To access the LEARN area for this coursework:

1. Log-in to LEARN at http://www.learn.ed.ac.uk OR via http://www.myed.ed.ac.uk, using your EASE username and password.
2. Click My Courses.
3. Click Programming Skills(2015-2016)[SS1-SEM1].
4. Click Coursework.

To download a file, right-click on the associated link and select Save As...

10 Appendix - how to create your own landscape

The LEARN area for this coursework (see Appendix 9) has a `pnm2dat` executable (and its source code). This converts PNM files, produced by `gimp` to to ASCII files.

You can create your own landscape as follows:

1. Start the image manipulation program `gimp`.
2. Select `File => New...`.
3. Choose the height and width (assuming one grid square = one pixel).
4. Click `Advanced Options`.
5. Select `Colour space: Greyscale`.
6. Click `OK`.
7. Draw your map.
8. Select `File => Export As...`
9. Enter a filename with an ending `.pnm` e.g. `file.pnm`
10. Click `Select File Type`
11. Select `PNM image`
12. `Export`.
13. Select Data Formatting: Ascii
14. Click `Export`.
15. Select `File->Quit` to exit

16. Run `pnm2dat file.pnm file.dat` to convert the PNM file to the same format as the supplied input file. Dark points (greyscale < 128) are considered land (represented as 1 in the `.dat` file, light points (greyscale ≥ 128) are water (represented as 0 in the `.dat` file).

To change brush size in `gimp`, select `Windows => Dockable Dialogs => Tool Options`.