**AdaptiveIDE User Guide**

**Authors:** Mark A. Gilbert, Steven M. White, James M. Bullock, Eamonn A. Gaffney

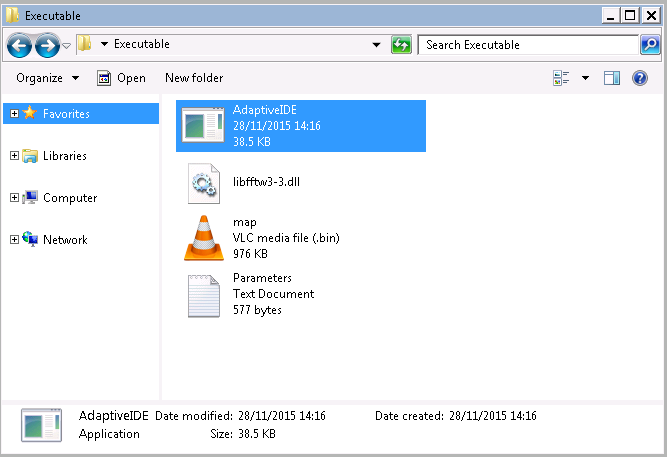
AdaptiveIDE was written in C++, compiled using Visual Studio 2010 and has been tested on a Windows 7PC. The steps below outline how to run the code.

Getting Started

To use AdaptiveIDE on Windows, simply download the zipped folder *AdaptiveIDE.* The folder contains four files that are necessary for the simulation (we can ignore all others for now):

* **AdaptiveIDE.exe** – The compiled programme
* **libfftw3-3.dll** – The Dynamic Link Library (dll) file for the Fastest Fourier Transform in the West (FFTW) library. This precompiled dll file was downloaded from the fftw website, <http://www.fftw.org/install/windows.html> . Without this file, AdaptiveIDE.exe will not be able to run.
* **map.bin** – a binary file containing a map of the landscape as boolean variables. 1s denote suitable habitat and 0s denote unsuitable habitat.
* **Parameters.txt** – a text file containing all the (changeable) parameters used in the model and in the simulation algorithm.

In this user guide, we will briefly describe what type of integro-difference equation AdpativeIDE solves in **What AdpativeIDE Does**, list the **Inputs**, explain **How to Run a Simple Example**, and finally explain **How to Run Your Own Examples**.



What AdaptiveIDE Does

AdaptiveIDE simulates integro-difference equations using adaptive mesh refinement. It simulates integro-difference equations of the form

where is the population distribution, is the spatially homogeneous dispersal kernel, is the simulation domain and is the density and location dependent growth function. In AdaptiveIDE, the choices of dispersal kernel are restricted to Tophat, Cone, Hemisphere, Gaussian and Laplace and the growth function is

Where is the carrying capacity (set to 1) and is the location dependent intrinsic growth rate. This is equal to (the intrinsic growth rate) in suitable habitat and zero in unsuitable habitat. The suitable and unsuitable habitat are defined by the binary file map.bin .

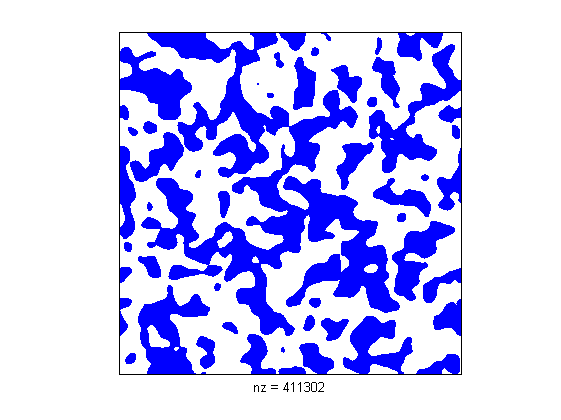
The initial population is constant and equal to 0.1 over a single region (the lower-resolution cells into which the landscape is partitioned).

Inputs

The zipped folder *AdaptiveIDE* contains four files. All the user-definable parameters are stored in Parameters.txt and a map of the landscape is stored in map.bin.

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| initial\_popn\_loc\_x | The x co-ordinate of the region containing the initial population |
| initial\_popn\_loc\_y | The y co-ordinate of the region containing the initial population |
| initial\_popn\_density | Initial population density |
| kernel\_no | Dispersal kernel (1 – Tophat, 2 – Cone, 3 – Hemisphere, 4 – Gaussian, 5 – Laplace) |
| length | Landscape length |
| print | Print to file? (yes – 1, no – 0) |
| intrinsic\_growth\_rate | Intrinsic growth rate in suitable habitat |
| output\_resolution | Resolution of output map |
| T | Number of time-steps |
| landscape\_width | Width of map in pixels |
| landscape\_length | Length of map in pixels |
| region\_width | Width of each region (in pixels) |
| threshcoarse | Population density threshold at which mesh returns to low resolution |
| threshcoarse2: | Population density change threshold at which mesh returns to low resolution (if density threshold *threshcoarse* has not been reached) |
| threshcoarse3 | Minimum population density that must be attained for mesh to return to low resolution |
| threshfine | Population density threshold at which mesh becomes high resolution |

The map file is by default the random map shown below. This cannot be read directly into MATLAB using load but requires the fread and fwrite commands (see the **How to Run Your Own Examples** section)



How to Run a Simple Example

To run AdaptiveIDE using the inputs in *Parameters.txt* and using the map in *map.bin*, just click on AdaptiveIDE.exe. Assuming that print=1 in *Parameters.txt*, then this will result in the generation of an output file *output.bin*, which contains maps of the population distribution at each time-step which are concatenated together (stacked on top of each other). To read them into MATLAB, use

>> load(‘PATH\output.bin’);

>>for t = 0:(T-1)

mesh(output((N\*t+1):N\*(t+1), : ));

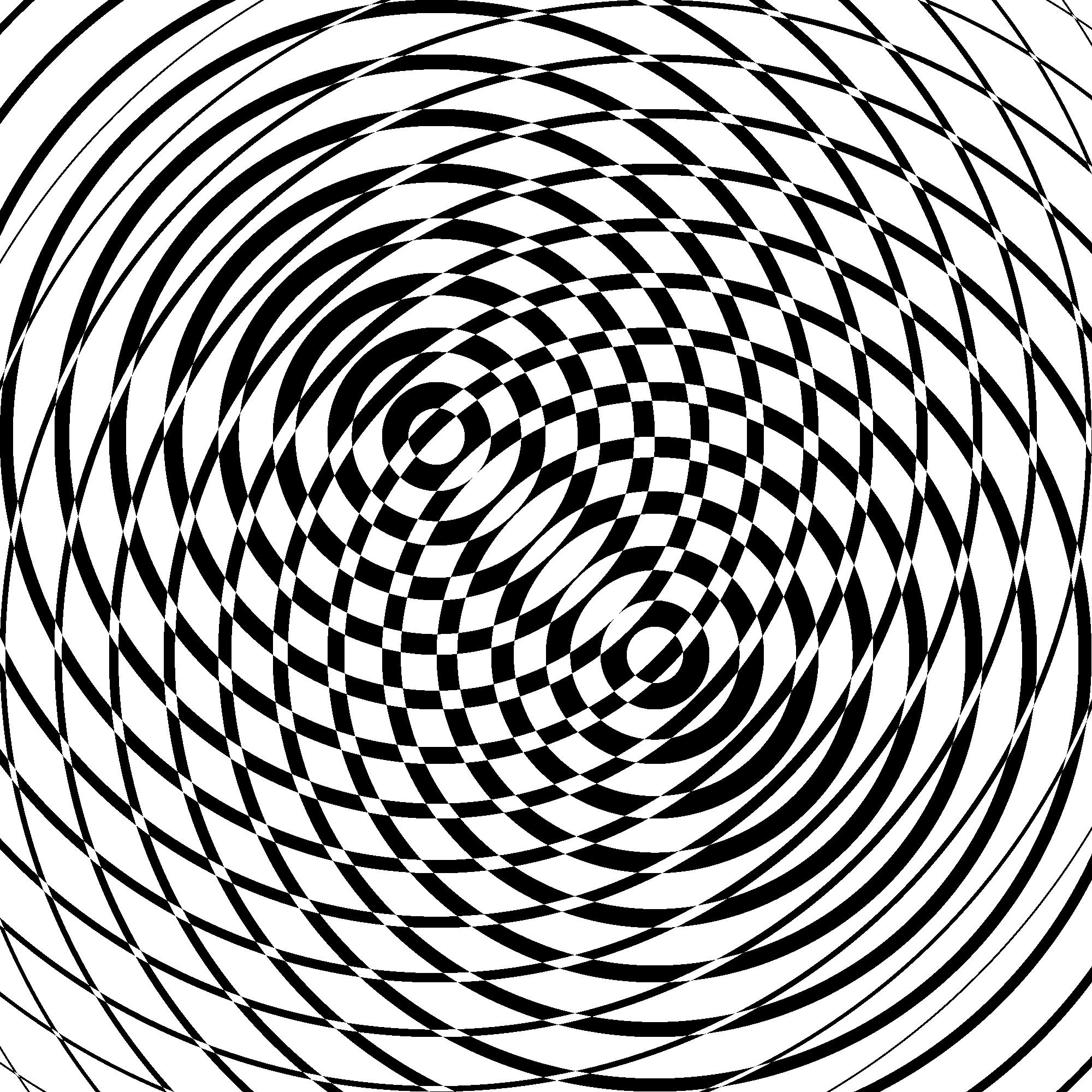
pause(.5);

end

where N is the height of the map in pixels and T is the number of time-steps.

How to Run Your Own Examples

To run your own example with a different map, e.g. the 2000 x 2000 concentric circle jpeg figure included with the code using a Gaussian dispersal kernel, with intrinsic growth rate equal to 20 and regions of size 200 x 200. We’ll take the map to be of an area of 200km x 200km and the dispersal parameter to be 3 km.



open the map image (e.g. jpeg) in MATLAB:

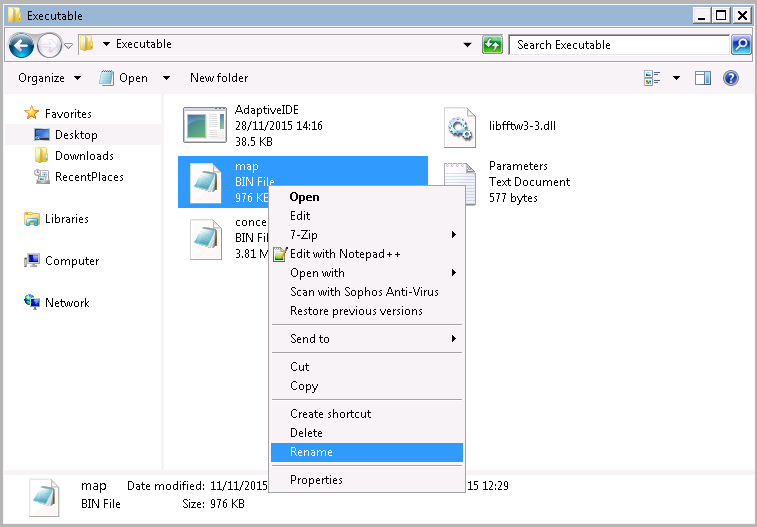
>> Z = imread(‘PATH\concentrics.jpeg');

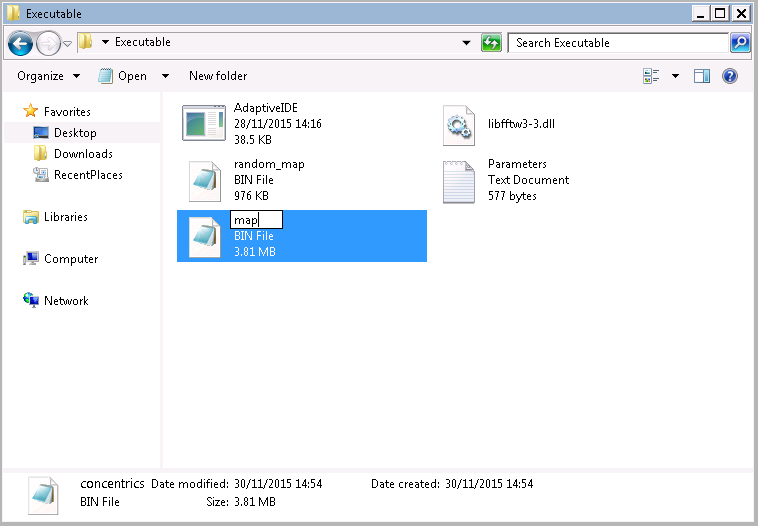
>> Z = Z>100;

>> fileID = fopen('PATH\concentrics.bin','w');

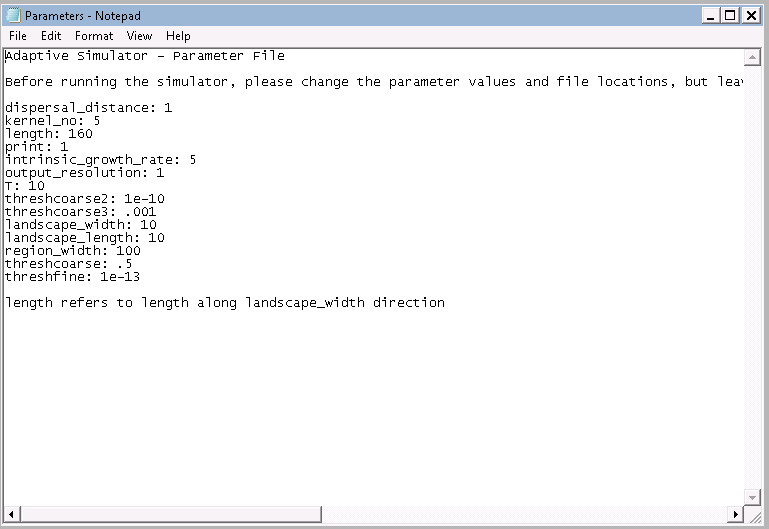
>> fwrite(fileID,Z,'bool');

This turns a black and white map into a Boolean binary file. This should output concentrics.bin into the folder containing the AdaptiveIDE code. To run a simulation using this map, we’ll need to rename concentrics.bin as map.bin. However to avoid losing the existing map, we should rename it as something else, e.g. random\_map.bin, and then rename concentric.bin as map.bin



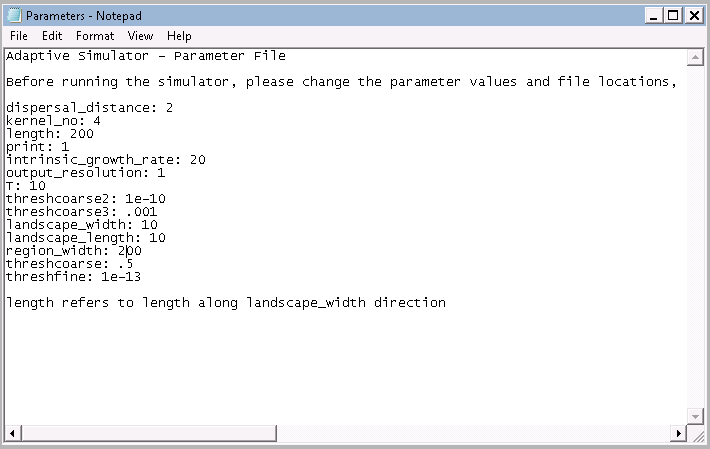


This changes the map. We now need to change the Parameters file. Simply open the Parameters.txt file in a text editor (e.g. notepad). When initially opened, the text file should look like this:



* Since the dispersal distance is 2km and the length of the landscape is 200km, we change the dispersal\_distance to 2 and length to 200.
* The dispersal kernel is now Gaussian (kernel\_no = 4), so we change the number after kernel\_no to 4.
* The intrinsic growth rate is now 20, so we change this.
* The width of each region in cells is now 200, so we change region\_width
* This means that the landscape is a 10 x 10 array of regions, so landscape\_width and landscape\_length stay as 10.

Changing these values (and saving the file) leaves us with a text file of the following form.



Before running the programme with the new parameters and map, make sure you save previous outputs as anything called output.bin in the file will be over-written. Once you’ve done this , you can run the programme with the new map and parameters by clicking on the executable. This should generate output.bin, which you can read into MATLAB (for details, see **How to Run a Simple Example)**. The map at the final time-step should look similar to the image below:

