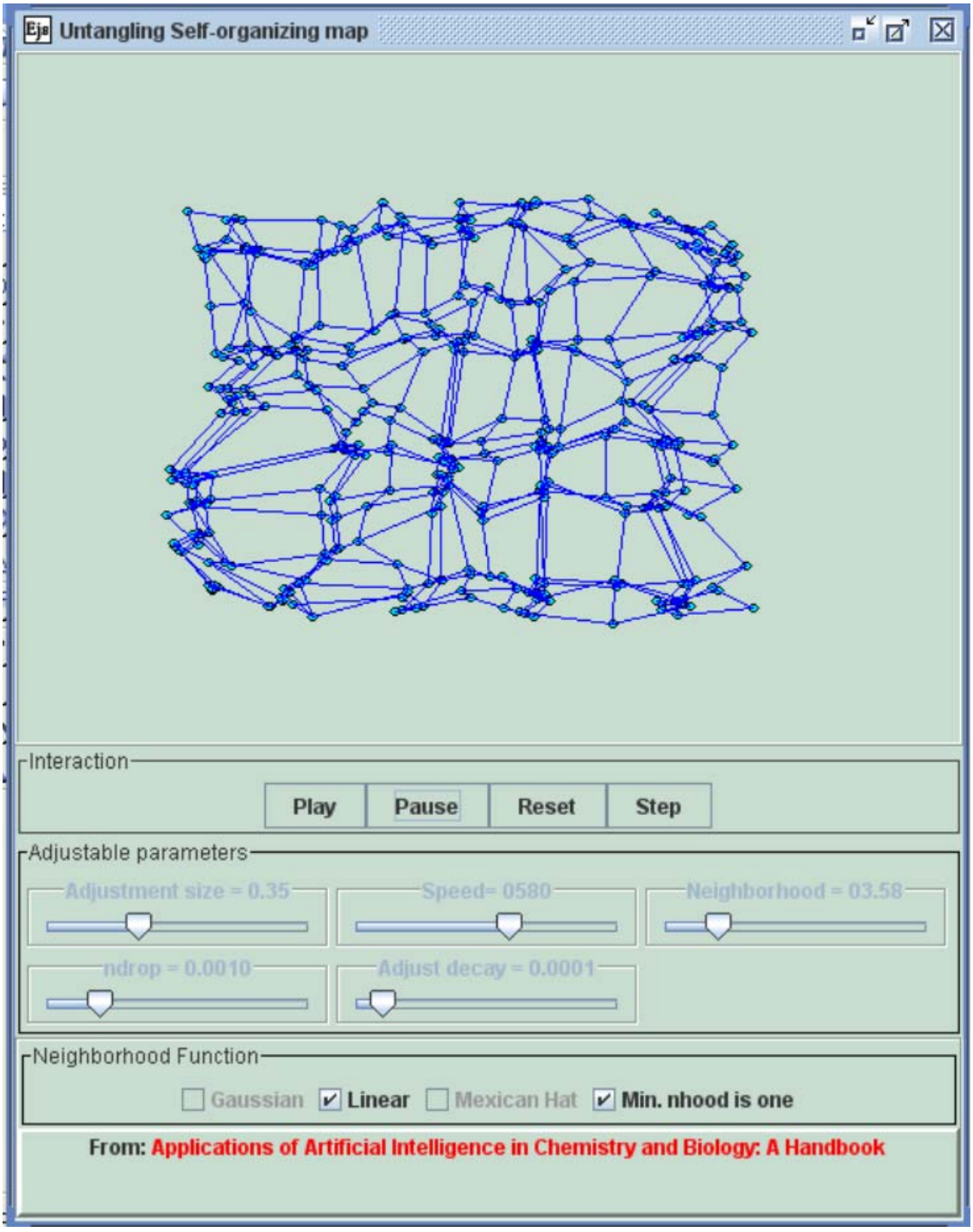
Self-organizing Map – Untangler

Input data to the SOM untangler consists of two-dimensional data points. Each point is picked at random from within the range 0 to (mapsize-1), where mapsize is the size of the two-dimensional map.



As the SOM runs, the pair of weights at each node gradually adjusts so that the entire set of nodes covers the range of input data values and nodes that are adjacent to one another develop similar weights. Lines drawn between nodes that are adjacent to one another in the node array thus show the network gradually untangling. This illustrates the way in which a SOM may adjust to reflect the geometry of the space that the input data define.

BUTTONS

Button	Function
Play	Restart the program if it has previously been paused.
Pause	Temporarily halt execution
Reset	Restart the calculation from scratch; all parameters are set to their default values.
Step	Execute a single cycle of the program.

SLIDERS

Slider	Default	Comment
Adjustment size	0.1	A proportionality factor that determines the size of the adjustment made to weights each cycle
Speed	1	Determines the frequency with which the display is updated. If updating is done less often, the program runs (much) faster.
Neighborhood	4	Current size of the neighborhood; maximum 18, minimum zero, unless the minimum neighborhood=1 box is ticked. The neighborhood is given as a real number, but the decimal part is ignored when updating the map. The decimal part is required because the neighborhood drops by a small fraction each cycle (as determined by ndrop).
Ndrop	0.001	The diminution of the neighborhood size each cycle.
Adjust decay	0.0001	The amount by which the adjustment factor for the weights drops each cycle.

Investigations and exercises

1. Default parameters

Run the simulation using the default settings. The size of the adjustments made to the weights each cycle and the size of the neighborhood both fall as the cycles pass and you will find that almost invariably the network fails to untangle before the size of the adjustment diminishes to a very small value. The default settings are therefore inappropriate.

2 Adjustable parameters

Restart the simulation and try adjusting the size of the neighborhood and/or the size of the adjustment per cycle to see what effect these have on the convergence of the map towards a stable state.

3. Mexican hat

The neighborhood function used is linear. If you know Java, try modifying the program, or writing your own program from scratch, to use the Mexican Hat function (formula in the book).

Note: **Adjust decay** determines how rapidly the adjustment size diminishes; **ndrop** determines how rapidly the neighborhood shrinks. **Speed** determines how often the display is updated; higher values of speed reduce the frequency with which the display is updated, and because drawing to the screen is a comparatively slow process, less frequent updating leads to more rapid execution of the algorithm.