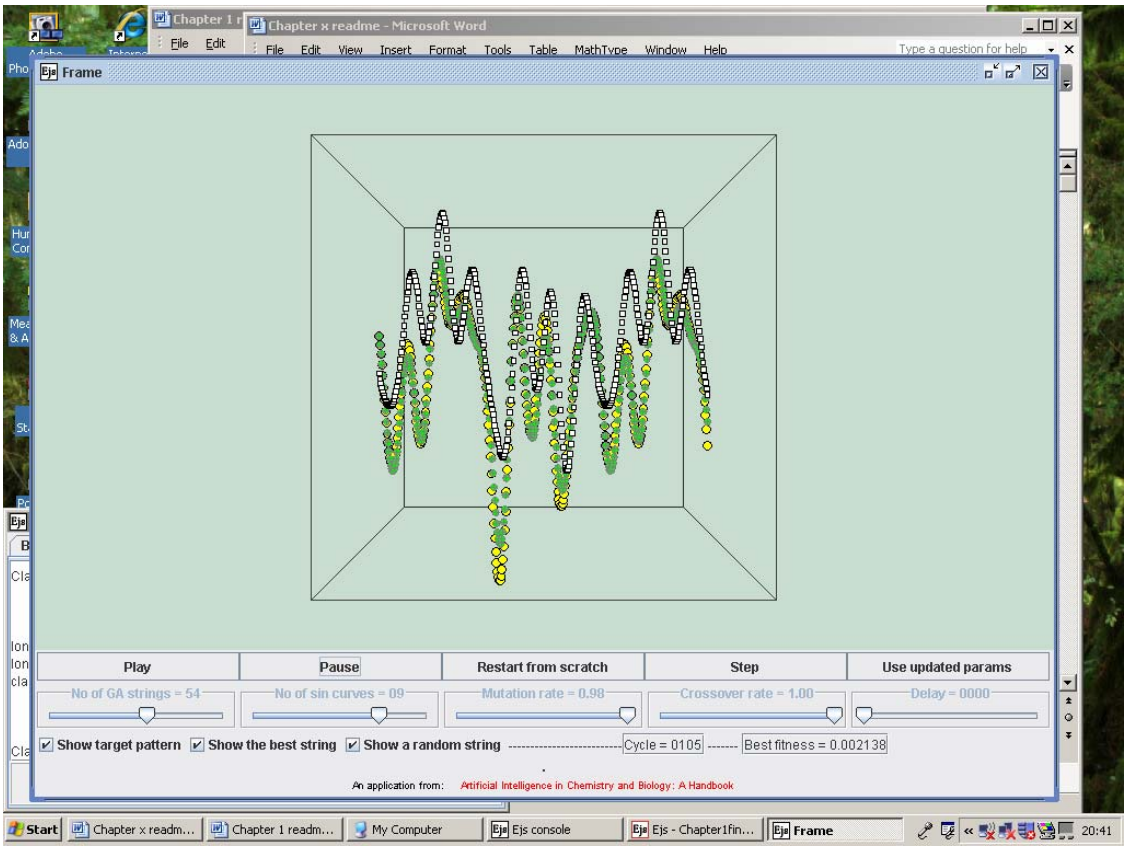
Genetic Algorithm Sin Curve Fitter

The simple Genetic Algorithm (GA) sin curve fitter can be run using Ejs (see the Ejs ReadMe file if you have not yet installed Ejs).



The program first generates a curve by selecting at random values for the parameters defining several sin waves:

```
for (icoeff=0; icoeff<nsincurves; icoeff++)
{
    xcoeff[icoeff]=15.0*Math.random();
    offset[icoeff]=15.0*Math.random();
}
```

then summing the resulting sin waves to give the target pattern, the yellow line in the figure above.

```
for (ipoints=0; ipoints<npoints; ipoints++)
{
    z[ipoints]=50.0;
    {
        for (j=0; j<nsincurves; j++)
z[ipoints]=z[ipoints]+xcoeff[j]*Math.sin(offset[j]+j*x[ipoi
nts]/10.0);
    }
}
```

The role of the GA is to find the set of coefficients and offsets which gives a matching curve, shown in green. The GA strings therefore consist of the values for these parameters, which it attempts to optimize. The best fitness of any string found in the current run is displayed, as is the number of cycles executed.

BUTTONS

The buttons in the window created by the program have the following functions:

Button	Function
Play	Restart the program if it has previously been paused.
Pause	Temporarily halt execution; execution is restarted using the **Play** button. (Note: it is easy to forget that the **Pause** button has been pressed. For example, if you **Pause** the run in order to change some parameters, then press the **Use updated params** button to start a calculation with the new values, the **Pause** will still be in effect; press **Play** to restart.)
Restart from scratch	Reset all parameters to their initial values and start the program from the beginning. (Since the program uses random numbers at several places in the calculation, repeated resetting will not lead to identical runs.)
Step	Run the program one cycle at a time.
Use updated params	Start the program from scratch, but use the latest values for parameters such as mutation rate and number of strings, as set using the sliders.

SLIDERS

The parameters that can be set by the user include:

Parameter	Default	Comment
No of GA strings	80	Variable between ywo and 96
No of sin curves	5	The curve which the GA is attempting to fit is the sum of between one and 12 sin curves.
Mutation rate	0.2	This is the GA mutation rate, which can be varied between 0 (no string is mutated) and 1 (every string is mutated once per cycle).
Crossover rate	1.0	This is the GA crossover rate, which can be varied between 0 - no string undergoes crossover - and 1 - the number of crossovers is half the number of strings each cycle, so on average each string undergoes crossover once per cycle. Note that in the latter case it is probable that some strings will undergo crossover more than once in a given cycle, while other strings may not be crossed.

Investigations & Exercises

1. Default parameters

Run the algorithm with the default settings. It should quickly settle on a good fit, but is unlikely to find a perfect fit. The fitness of a string is calculated by the following line in the program

```
fitness[istring]=10.0/(1.0+lssum);
```

in which lssum is the sum of the squared deviations of the calculated points from the target points. It follows that the maximum theoretical fitness is 10.0. No run of the program is likely to generate a string of this fitness, since this would require that the program find the exact floating point values which have been chosen at random to define the target curve.

2. No mutation

Now try running the algorithm with a mutation rate of zero (use the mutation slider to set the mutation rate to zero, then click on **Use updated params**). The fit should quickly improve from its starting point, but will soon settle down and all further improvement will cease, as can be shown by clicking the **Show a random string** tick box. This is because in the absence of mutation crossover will ensure that every GA string eventually becomes identical, so evolution comes to a halt.

3. No crossover

Set the crossover rate to zero and the mutation rate to one, then click on **Use updated params**. The best string may improve indefinitely, but only sporadically, since most mutations will be ineffective and once again the population will quickly become filled with almost identical strings. In this run however, mutation continues to operate, so new strings are repeatedly generated and will occasionally result in an improved string. Note that exercises 2 and 3 show that both the mutation and the crossover operators are generally required for a GA to function effectively.

4. Population size

Also investigate how changing the number of strings from a small value to a large one affects (a) the speed with which a good string is found, and (b) the quality of the best string found after, say, 100 generations.

5. Effect of mutation

Try displaying a random string as well as with the best one each generation. The best and the target curves will soon be very similar if you have chosen reasonable values for the number of strings, mutation and crossover rates. Notice that, as the calculation converges, a string chosen at random is usually similar to the best string each generation, but that occasionally, as a result of an unfavorable mutation, the random string is very different from the best string.