WINTER EXAMINATION 2020

M.Sc. in Applied Science (Mathematical Modelling and Self-learning Systems)

APPLIED MATHEMATICS

AM6007 – Scientific Computing with Numerical Examples

Dr Kieran F. Mulchrone

ATTEMPT ALL QUESTIONS

PREDATOR-PREY DYNAMICS

In mathematical biology there is much interest in modelling the behaviour of predator-prey populations with obvious applications in harvesting. One continuous model, which comprises a system of two ordinary differential equations, is given as follows:

$$\frac{dx}{dt} = rx(1-x) - bxy$$

$$\frac{dy}{dt} = (-d + bx)y$$

where x and y denote prey and predator population densities respectively. The system parameters (r, b, d) are all positive constants. This system of equations can be discretised as follows:

$$x_{n+1} = x_n + \delta(rx_n(1-x_n) - bx_n y_n)$$

$$y_{n+1} = y_n + \delta\big((-d + bx_n)y_n\big)$$

where δ is a typically small positive constant. Hence given some initial values(x_0, y_0), these equations can be iterated to find a sequence of future values.

We could also think of these equations in terms of vectors of values and a vectors of functions. Let the vector of values $(\boldsymbol{v}_n)$ be:

$$\boldsymbol{v}_n = \begin{pmatrix} x_n \\ y_n \end{pmatrix}$$

Let the vector of functions $(\boldsymbol{f}(\boldsymbol{v}_n))$ be:

$$\boldsymbol{f}(\boldsymbol{v}_n) = \begin{pmatrix} rx_n(1-x_n) - bx_n y_n \\ (-d + bx_n)y_n \end{pmatrix}$$

Thus the above system of equations may also be written as:

$$\boldsymbol{v}_{n+1} = \boldsymbol{v}_n + \delta\, \boldsymbol{f}(\boldsymbol{v}_n)$$

In this question you must produce software to generate data which can be used to produce bifurcation diagrams for the discrete system. A bifurcation diagram is produced as follows:

1) Choose a set of parameters (r, b, d) and an initial value (x_0, y_0).
2) Choose a minimum and maximum value for δ and subdivide the interval into however many subintervals you require.
3) For each value of δ iterate the equations for N_{settle} times – this allows transient behaviours to settle down. Then for N_{reps} iterations write the values of (δ, x_n, y_n) to a comma separated value file format.
4) Produce a scatter plot of δ vs. x_n (i.e. for all values of δ).

PART A (25 Marks)

Create a vector class, which encapsulates an array of double values, as follows.

Data

private double[] values; this is the array of doubles.

Methods

public Vector() : sizes the array to be of length 2 by default.

public Vector(int size) : sizes the array to be of length size but defaults to 2 if the input is erroneous.

public Vector(double[] values) : sizes the array according to the input array of doubles and copies the values to the private array in the object.

public void setSize(int size) : resizes the array if necessary and copies as many of the existing values as possible into the newly sized array.

public static Vector operator +(Vector lhs, Vector rhs) : overloads the addition operator

public static Vector operator -(Vector lhs, Vector rhs) : overloads the subtraction operator

public static Vector operator *(double lhs, Vector rhs) : overloads double by vector multiplication

public static Vector operator *(Vector lhs, double rhs) : overloads vector by double multiplication

public double this[int index] : an indexer method to allow the user to get set the elements of the array.

public override string ToString() : outputs the contents of the vector in string form, for example “{1, 2, 3}”.

PART B (25 Marks)

Create a delegate called Function which can reference methods that take a vector as an argument and returns a double. (5 Marks)

Create a FunctionVector class which encapsulates an array of delegates of type Function as follows. (20 Marks)

Data

private Function[] functionVector : the array of delegates of type Function.

Methods

public FunctionVector() : size the array of delegates to 2 and provides default lambda expressions which return zero (e.g. x => 0).

public FunctionVector(int size) : size the array of delegates to size (or default to 2 if input is erroneous) and provides default lambda expressions which return zero.

public FunctionVector(Function[] functions) : size the array of delegates according to the size of functions. Make the array of delegates reference the delegates in functions.

public Vector Evaluate(Vector values) : evaluate the delegate array at the vector values and return a vector of doubles containing the result of the evaluation.

public Function this[int index] : an indexer method to allow users the get and set delegates to individual elements of the array.

Test your implementation of Vector and FunctionVector with the following code:

```
//test of Vector and FunctionVector
FunctionVector v = new FunctionVector(3);
v[0] = x => x[0] + x[1] + x[2];
v[1] = x => x[0] + x[1] * x[2];
v[2] = x => x[0] * x[1] + x[2];
Vector tmp = v.Evaluate(new Vector(new double[] {1,-1,3}));

Console.WriteLine("tmp = {0}", tmp);
```

PART C (50 Marks)

Create a class PredPrey which generates the data to produce a bifurcation diagram for the discrete predator-prey system described earlier in this paper.

Data

private FunctionVector fv; : vector of delegates of type Function

private double delta, r, b, d; : system parameters (corresponding to equations in predator prey system). Default values are delta = 0.5, r = 2, b = 0.6, d = 0.5. Provide a property for each parameter.

private int nsettle; : number of iterations to let the system settle down. Default value 200. Provide a property with get/set methods such that the minimum value is 10.

private int nreps; : number of iterations to iterate the system and record the results. Default value 200. Provide a property with get/set methods such that the minimum value is 10.

Methods

public PredPrey() : in the constructor set up functionvector to refer to lambda expressions encapsulating the discrete system discussed earlier in the paper.

public void runsimDrange(Vector v0, double deltafrom, double deltato, int numsteps, string filename) : given an initial value v0, run a simulation for values of delta (i.e. δ) ranging from deltafrom to deltato in numsteps. For example, if deltafrom = 0.5 and deltato = 1.0 and numsteps = 3 a simulation would run for delta = 0.5, 0.75 and 1.0. The filename gives the file where the data are recorded. Make use of run1sim() in this method.

public void run1sim(Vector v0, string filename) : This iterates the equations with initial value v0 first for nsettle times and then for nreps times. Write the iteration using the Vector and FunctionVector classes created earlier. The results are written for nreps times to the file referred to by filename in comma separated variable format as follows:

"delta, xn, yn"

For example:

1.379,0.85680980672395,0.123700725536175

1.379,1.10748542899256,0.126103542125421

1.379,0.663624224865903,0.154708042879828

1.379,1.19433709525472,0.132984354681508

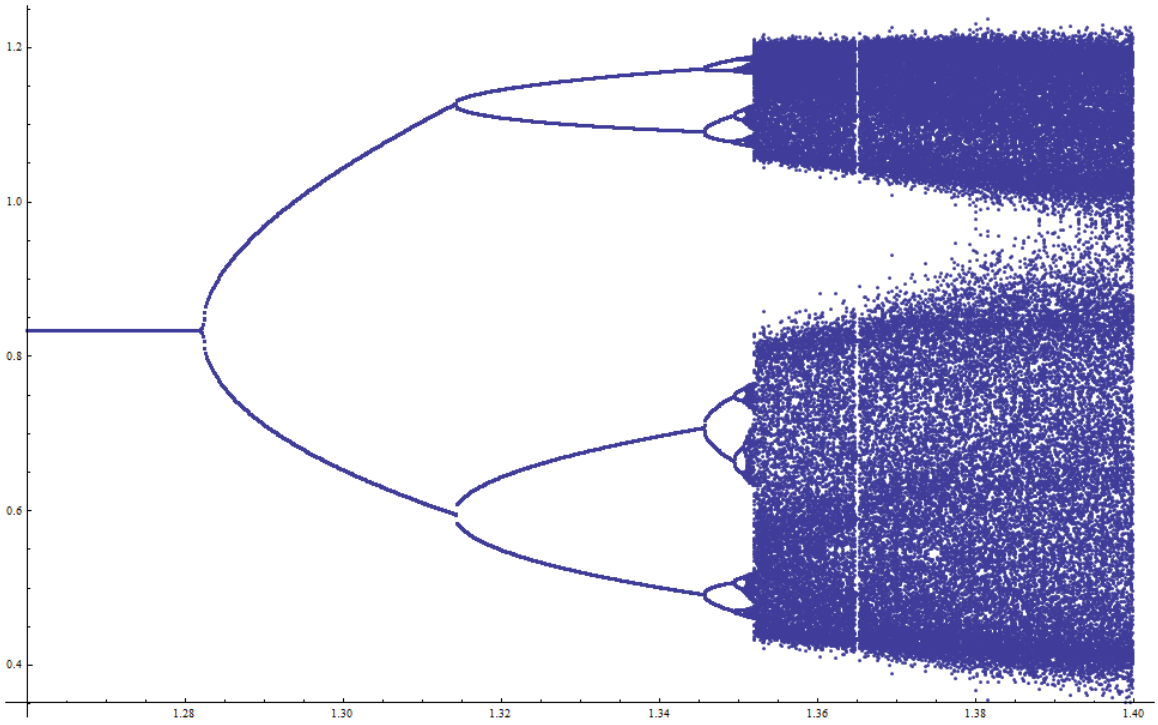
Make sure you are appending the data to the end of the file each time you write to the file, e.g. if you use the StreamWriter class it is an option in the constructor.

Test your implementation with the following code (change file names and locations as necessary):

```
//predator prey simulation
//(1) single value
PredPrey p = new PredPrey();
p.Nsettle = 1000;
Vector v0 = new Vector(new double [] {0.83,0.55});
p.Delta = 1.38; //this uses the current value of Delta.
p.run1sim(v0, "c:\\users\\km\\outfile.csv");

//(2) produce bifurcation plot data use default values
p.runsimDrange(v0, 1.26, 1.4, 1000, "c:\\users\\km\\outfile1.csv");

//(3) produce second bifurcation plot
p.R = 3;
p.B = 3.5;
p.D = 2;
v0 = new Vector(new double[] { 0.57, 0.37 });
p.runsimDrange(v0, 0.5, 0.95, 1000, "c:\\users\\km\\outfile2.csv");
```

The following code in mathematica was used to create the pretty pictures:

```
data1 = Import["c:\\users\\km\\outfile1.csv"];

ListPlot[Take[data1, All, {1, 2}]]
```



As part of your submission please provide one such image (you can use whichever software you like to produce the image).

PLACE ALL YOUR CODE INTO A SINGLE "*.CS" FILE FOR SUBMISSION.

SUBMIT THE CODE FILE AND ONE IMAGE FILE.