

MATLAB EXERCISES

ADAM ATTARIAN

- (1) We're going to work with some simple population dynamics in this problem. Consider the Lotka-Volterra population model, given by

$$\begin{aligned} (1) \quad & \dot{x} = x(\alpha - \beta y) \\ (2) \quad & \dot{y} = y(-\gamma + \delta x), \end{aligned}$$

where α, β, γ , and $\delta > 0$. In the equations, $x(t)$ represents a prey population with positive birth rate α , and $y(t)$ are the predators which dies off if its food source is absent.

- (a) Solve the model numerically using one of the Matlab ODE solvers over the time interval $0 \leq t \leq 10$. Try to find some parameters that give interesting dynamics. We'll call these parameters $q^* = [\alpha^*, \beta^*, \gamma^*, \delta^*]$.

Plot both the time series (that is, the solutions against time) as well as the phase portrait (one state against the other). An example is in Figure 1.

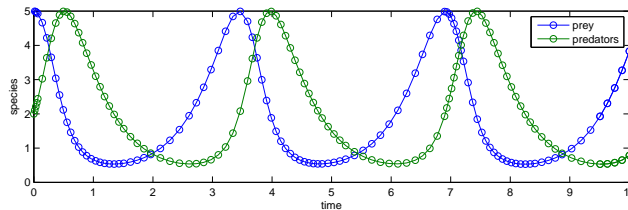


FIGURE 1.

- (b) Once you have nice looking results, save the results using the `save` command. We'll treat this as collected data.

We're going to try to estimate the parameters α, β, γ , and δ from the “data” that has been collected. Write a function that given a parameter vector, solves the model and compares these solutions to the “data”. The function should return the sum of the errors squared, a scalar. You can use the same initial conditions that you used to generate the data, though keep in mind normally you wouldn't know the initial conditions.

- (c) Once you have a function that returns a scalar given a vector input, you can minimize the function using one of the algorithms from the Optimization Toolbox. Do this using `fminsearch` and/or `fminunc`. Use an initial iterate that is not equal to q^* . Try several initial iterates and see what you find. This is a “zero residual problem”, so you should

be able to exactly reproduce your original parameter that the data were generated with.

- (2) If you've gotten this far, congratulations. Now we're going to play a chaos game in the process generating the Sierpinski Triangle, a nice fractal. The game is played as follows.

Start out with a unit equilateral triangle, with vertices at $(0, 0)$, $(1, 0)$, $(.5, \sqrt{3}/2)$. Given a starting point within the triangle (x, y) , *randomly* choose one of the vertices. Take the midpoint of the line joining this vertex with the starting point and plot this new point. Then we take the midpoint of the line joining this point and a randomly chosen vertex as the next point, which is plotted, and we continue doing this for a large number of times.

So you're plotting each new midpoint as we go along. Some commands that would be helpful in this are `rand`, which gives you a random number between 0 and 1, `ceil` which rounds towards ∞ , and `hold on` which will help you plot each successive point with out clearing your plot. Bonus points if you can make each point a random color, making a very colorful triangle in the process. For $N = 1000$ I get the picture in Figure 2.

- (3) Do this problem only after we've talked about sensitivities and subset selection.
- (a) Using `tssolve.m`, compute the relative and regular sensitivities of the population model.
 - (b) Compute the rankings of the parameters. For your parameter set, which is most sensitive? Does this make sense?
 - (c) Perform the QR with column pivoting on the sensitivity matrix assuming you can measure both populations. What does this analysis say is the most identifiable?
 - (d) Compute the eigenvalues of the sensitivity matrix. The largest eigenvalue corresponds to the parameter that the QR analysis said was most identifiable. What is the range of the eigenvalues?

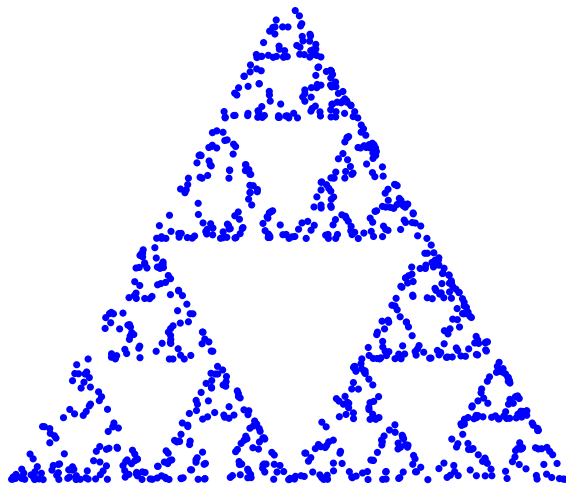


FIGURE 2.