MATH 308 - SECTIONS 503/504
PROJECT 3 - THE ZOMBIE APOCALYPSE!

INSTRUCTOR: DR. ADAM LARIOS

Due date: Monday, 3 Dec. 2012, in class

1. Introduction

In the paper, "When zombies attack! Mathematical modeling of an outbreak of zombie infection" [Munz, Hudea, Imad, and Smith, 2009], the authors analyze an outbreak of zombiism as an infectious disease and discover that, in their models, the only stable equilibrium is one in which all humans are turned into zombies. This is called the Zombie Apocalypse. Your mission is to read their paper and question their assumptions. Can you find an incorrect assumption? How do you change the model for this? Can you find a stable equilibrium? You may also think of other ways that humans might remove dangerous zombies from the population. Does this change the model? In what ways? And does this allow for a stable equilibrium? You may want to do a web search and/or a library search to see if you can find other publications modeling disease outbreaks, and see what conclusions those come to. You should learn some basics about disease modeling, at the very least reading about it on Wikipedia [Wikipedia(2011)].

Your main goals are:

- Modify your Runge-Kutta-4 code from Project 1 (or modify my example on the webpage) to allow for 2D systems. Verify that your code is correct by checking that your error is order $\mathcal{O}(h^{-4})$ as you did on Project 1. Come up with your own, ODE system for validation purposes, for which you know the exact solution (we did something similar in Project 1).
- Reproduce some of the results from the [Munz, Hudea, Imad, and Smith, 2009] paper.
- Once you have this working, evaluate and change the modeling assumptions.
- Write equations to reflect your changes, and see how this influences the equilibria.
- See if you can figure out what are the best ways for humans to fight the zombies and prevent the apocalypse.

2. Instructions

Your job is to explain this topic and your modeling in your own words. If you use any references to review the topics, you should cite these references in your bibliography. If you quote from a reference, you must use quotes and a citation. If you paraphrase closely or take a problem from a reference, you use a citation. Avoid excessive quotation and paraphrasing. To accomplish this, as much as possible do not look directly at a reference while writing up your explanation, so that you use your own words. If you need to, make an outline of the major ideas and then refer back to your outline while you are writing.

Your write-up should include:

(1) An introduction. In the introduction you should:
   (a) Explain your understanding of zombies and zombiism (with references to your sources, as in the original paper).
   (b) Discuss the prior paper, and its results.
(2) A modeling section in which you explain your new modeling assumptions (similar to section 2, 3, 4 of the original paper), and your results.
(3) You may have two or three subsections in which you explore alternatives (similar to the original paper). In particular, you should take into account what actions the humans might take.
(4) In the modeling section or subsections, you should include graphs to show how the human and zombie populations change over time, a calculation of the equilibria, and an analysis of how the eigenvalues of the linearized system near the equilibria influence the behavior of the system.

   A results and conclusion section in which you include
   (a) An overall summary of your results.
   (b) An evaluation of which model(s) created the best scenarios for the humans. Are there any ways to avoid the apocalypse?
   (c) Any criticisms of your models that you can see.
   (d) A concluding paragraph that sums up the paper.
(5) A bibliography of any sources of information you cited.

You are invited and encouraged to have as much fun with this topic as you can!

- 15 points for the 2D Runge-Kutta code and an error check (i.e., slope of -4 on a log-log plot).
- 20 points for the introduction, along with correct and complete scientific/mathematical explanation of the general problem and equations, and data.
- 2×25 points for each section on a model (assuming 2 sections)
- 15 points for conclusions.

Total: 100 points.

Things to check before handing in your project:

(1) Have you done everything you were asked to do?
(2) Is the math right? If not, get it fixed.
(3) Is what you are trying to explain clear?
(4) Have you used imprecise language, speculative claims without justification, "weasel words," etc.? If so, clean these up.
(5) Do your graphs clearly show the conclusions you reach?
(6) Are the spelling, grammar, and English usage correct and concise?
(7) Do you think this is A, B or C level work?
(8) Run the document through a spell checker!
(9) Properly indent your code using Matlab's CTRL+A, CTRL+I functionality!
(10) Have the somebody read your document and critique it; then edit your document taking those comments into account.

3. Additional Information and Notices

While there are no specific neatness requirements, your work should look professional, and you should submit it in a form that would be appropriate for submitting to a boss at a job you care about. Credit will be given for full, complete, and thoughtful analysis; however, a good report is not necessarily a long report. Do not submit fluff or filler, but only quality work that you are proud of.

You are free to choose your partner, so long as it is a different partner than in previous projects. Submit it with all group member's names clearly labeled on the front. Make sure graphs are well labeled and referenced, and that they are easy to read. Please note that each person in a group is responsible for 100% of the work, so it is your responsibility to keep all group members on task, or to finish the work yourself. Submit all of your code, along with any relevant graphs, mathematical calculations, and explanations. You are to use pair programming techniques to do the Matlab. You are to split the writing tasks and critique each others work, bringing everything together into one coherent report at the end.

If a group member is not responding or falls out of contact, you need to let me know as soon as possible. If you are stuck on something in Matlab and are spinning your wheels, please email me early so you can get help when you need it, even if you don;t know what question to ask. This project is not demanding, but if you wait until the last week to start it, it is unlikely that you will do well. If you start it early, it should be very manageable.

By turning in this project, you agree to follow the Aggie Honor Code. In particular, please be sure any code you represent as your own is 100% written by you and your group members. You are free to discuss with other groups, but your code should be your group's own distinct code.

4. Appendix: Runge-Kutta Algorithm for a 2×2 System

Consider the initial-value problem,

$$\begin{cases} \dfrac{d}{dt}x = f_1(t,x,y), \\ \dfrac{d}{dt}y = f_2(t,x,y), \\ x(t_0) = x_{\text{initial}}, \\ y(t_0) = y_{\text{initial}}. \end{cases}$$

The solution for this 2×2 system will of course be two functions, $x(t)$ and $y(t)$. To solve it with Runge-Kutta-4, start with inputs:

$$x_0 = x_{\text{initial}}, \qquad y_0 = y_{\text{initial}}, \qquad t = t_0.$$

Then, update each time-step using

$$\begin{aligned} k_{1,1} &= h \cdot f_1(t, x_n, y_n); \\ k_{1,2} &= h \cdot f_2(t, x_n, y_n); \\ k_{2,1} &= h \cdot f_1(t + \tfrac{h}{2}, x_n + \tfrac{1}{2}k_{1,1}, y_n + \tfrac{1}{2}k_{1,2}); \\ k_{2,2} &= h \cdot f_2(t + \tfrac{h}{2}, x_n + \tfrac{1}{2}k_{1,1}, y_n + \tfrac{1}{2}k_{1,2})); \\ k_{3,1} &= h \cdot f_1(t + \tfrac{h}{2}, x_n + \tfrac{1}{2}k_{2,1}, y_n + \tfrac{1}{2}k_{2,2})); \\ k_{3,2} &= h \cdot f_2(t + \tfrac{h}{2}, x_n + \tfrac{1}{2}k_{2,1}, y_n + \tfrac{1}{2}k_{2,2})); \\ k_{4,1} &= h \cdot f_1(t + h, x_n + k_{3,1}, y_n + k_{3,2}); \\ k_{4,2} &= h \cdot f_2(t + h, x_n + k_{3,1}, y_n + k_{3,2}); \\ t &= t + h; \\ x_{n+1} &= x_n + (k_{1,1} + 2k_{2,1} + 2k_{3,1} + k_{4,1})/6; \\ y_{n+1} &= y_n + (k_{1,2} + 2k_{2,2} + 2k_{3,2} + k_{4,2})/6; \end{aligned}$$

Note that changing an existing 1D RK4 code into a 2D RK4 code only requires a slight modification, and should only take you a few minutes to reprogram. To test the global error, you want to test both x and y values together. Do this, for example, in by setting your error (for a given number of grid-points) as

$$\texttt{error} = \texttt{max}\left(((x_{\text{exact}} - x_{\text{approx}})^2 + (y_{\text{exact}} - y_{\text{approx}})^2)^{1/2} \right);$$

where x_{exact}, y_{exact} are vectors made by plugging all your t values into the exact solution you found for the test-problem you made up, and x_{approx}, y_{approx} are vectors which hold the calculated data from every time step. Of course, in Matlab, you will have to use the $.\wedge$ notation to square the vectors component-wise.