UECM3033 Assignment #3 Report

========================================================

- Prepared by: Kok Wen Jing

- Tutorial Group: T2

------------------------------------------------------------------------------------------------------------

## **Task 1** -- Gauss-Legendre formula

The reports, codes and supporting documents are to be uploaded to Github at:

[https://github.com/wenjingkok/UECM3033\_assign3](<https://github.com/wenjingkok/UECM3033_assign3>)

***Explain how you implement your `task1.py` here.***

We find the integration using Gauss-Legendre quadrature.

The equation for Gauss-Legendre quadrature is defined as below.

where = the weights in Gauss-Legendre quadrature,

= the nodes in Gauss-Legendre quadrature,

= the equation which substituted the respecting nodes,

= the number of points for the Gauss-Legendre quadrature.

After we get the weights and nodes in Python, we perform summation and translate the values of and the final answer to the interval we predefined since the weights and nodes values we found are based on the default interval which is from -1 to 1.

The following formulas are using to translate the value of and the final answer.

where is the translated values.

***Explain how you get the weights and nodes used in the Gauss-Legendre quadrature.***

We get the weights and nodes using the predefined function in Python.

The predefined function is the *leggauss* function found in *numpy.polynomial.legendre.*

The *leggauss* function need an input value which is the number of points for the Gauss-Legendre quadrature which is the value we defined at above.

The *leggauss* function return two arrays which are the nodes and weights for the -points Gauss-Legendre quadrature.

------------------------------------------------------------------------------------------------------------

## **Task 2** -- Predator-prey model

***Explain how you implement your `task2.py` here, especially how to use `odeint`.***

The *odeint* function required a predefined function in the Python module *scipy.integrate* that helps in solving ordinary differential equations with initial value problem.

The equations we used in this task is the transformed equations defined as below.

where and are constants, , , is the number of prey while is the number of predator.

The initial value we use for and are 0.1 and 1.0 respectively where we defined as below.

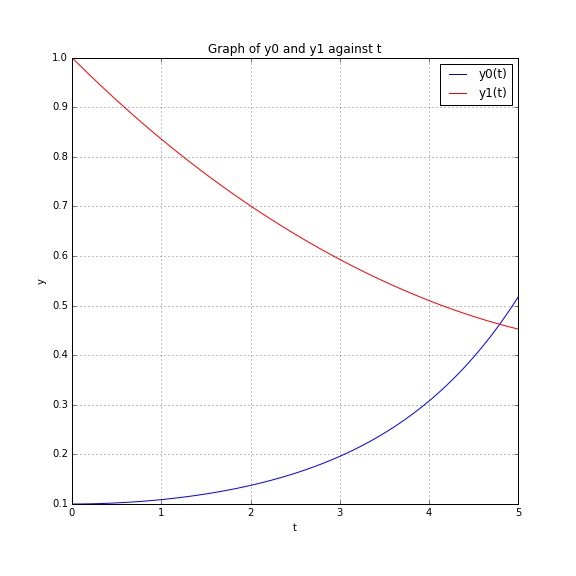
The above differential equations defined in a function named *deriv()* together with the initial value of and .

We solve the ODE problem through a time period of year, from 0 to 5 and divided the time period into 100 points. The time is defined as t = np.linspace(0, 5, 100) where the *linspace()* function will divided the interval [0,5] into 100 points. We get the derivative solution using the *odeint()* function as mentioned above where we called the function by below.

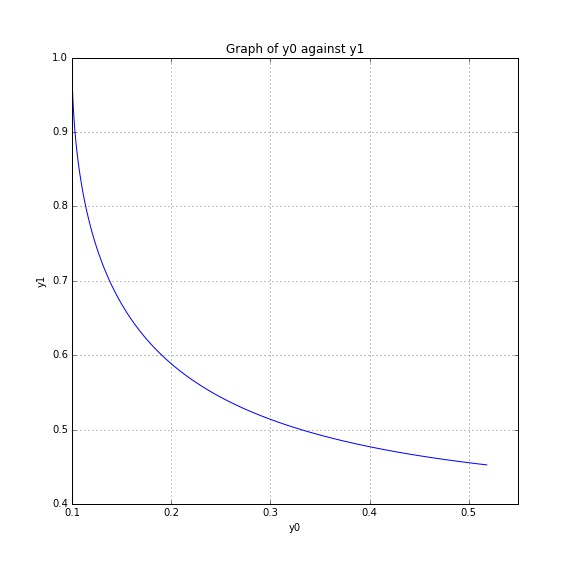
solution = odeint(deriv, y, t, args(a, b))

After get the solution from *odeint()*, we plot the following graph.

***Put your graphs here and explain.***



As time increase, the number of prey increasing while the number of predator decreasing.



There is an inverse relationship between the number of prey and predator.

***Is the system of ODE sensitive to initial condition? Explain.***

This system of ODE is not sensitive to initial condition since the changes of initial value does not make much different for the solutions we get.

------------------------------------------------------------------------------------------------------------

Last modified: 17 April 2016