ENGSCI331 ODE Name:____

Coding Improved Euler and the classical fourth-order Runge-Kutta methods

Correct implementation of `improved_euler` method	+1
Evidence of verifying a single step of the `improved_euler` method	+1
Correct implementation of the `runge_kutta` method	+2
Evidence of verifying a single step of the `runge_kutta` method	+1
`ode_main` neat, tidy, readable and sensible	+1
Incorrect implementation of a predictor or corrector step in either method	-1/2

/6

Sample function

Plots are present and qualitatively correct	+1

Predator-prey model

Correct implementation of the predator prey function	+2
Plots are present and qualitatively correct	+1

Bungee-jumper model

Correct implementation of the bungee-jumper function	+2
Plots present and qualitatively correct	+1

/7

Adaptive step-sizing

Code of an adaptive method that works	+2
Explanation in the report or comments of the theory	+1
Score adjusted for class rank	+2

If you have not coded a method that changes in step-size you will get no marks for this section.

/5

General coding considerations

Code should be neatly laid out, indented, consistently formatted, not unnecessarily complex, and a good expression of the programmer's intent. Clear expressions of intent make for code that is easy to understand, maintain, and debug. Code should be documented sensibly, using comments as well as sensible choices for variable names and routines. Note that good choices for variable names make your code "self-maintaining" as the comments or documentation are never out of step with the code itself – you can see from the code what it is doing at all times. Additional comments are still needed for pointing out important parts of the algorithm, making sense of complicated indexing or steps, and calling attention to things that might go wrong. All this makes beautiful code.

Beautiful code	+2

/2

Half a mark will be deducted once for each of the following: -1/2

Not using dynamic memory allocation appropriately, not checking for failure scenarios, or checking error codes of routines which return a non-zero value, hard coding data rather than using macros, unnecessary changes to interfaces, direct comparison of floating point values, poor documentation, code that doesn't do what you say it does, unnecessary tests within loops, unnecessary code duplication, anything else deemed bad code!

Report

The report should be targeted at a reader that has a complete understanding of the methods and techniques but does not know what you have been asked to do. The report should outline what you have been asked to do, detail anything interesting, unexpected, or complicated in the implementation; present and discuss the results; and present any suitable conclusions if appropriate.

If it is clearly presented, understandable and fits together well	+5
OR if the report shows interesting material but is not altogether clear or concise	+3
OR if the report shows some investigation and work but is poorly presented	+1
OR if no report or no effort has been made, no marks will be given	0

/5

/25