AME 20214
Homework 10
Due: Thursday, 15 November 2012, in class

Throughout the term, we have been numerically solving ordinary differential equations, a critical predictive tool used widely in engineering. We developed the first and second order Runge-Kutta methods; in actuality a first order Runge-Kutta method is simply the forward Euler method. Let us consider here the widely used *fourth order Runge-Kutta method.* Let us say we are solving N non-linear ordinary differential equations of the form

$$\begin{aligned} \frac{dy_i}{dt} &= f_i(y_j), \qquad i, j = 1, \ldots, N, \\ y_i(t_o) &= y_{io}. \end{aligned}$$

Then the fourth order Runge-Kutta algorithm to predict a solution at $t = t_o + \Delta t$ is as follows:

$$\begin{aligned} k_{1i} &= \Delta t f_i(y_j^n), \\ k_{2i} &= \Delta t f_i\left(y_j^n + \frac{k_{1j}}{2}\right), \\ k_{3i} &= \Delta t f_i\left(y_j^n + \frac{k_{2j}}{2}\right), \\ k_{4i} &= \Delta t f_i(y_j^n + k_{3j}), \\ y_i^{n+1} &= y_i^n + \frac{1}{6}(k_{1i} + 2k_{2i} + 2k_{3i} + k_{4i}) \end{aligned}$$

1. Build a `subroutine ode4`, similar in structure to `subroutine ode1` or `subroutine ode2` of the course notes, which implements the fourth order Runge-Kutta algorithm.

2. Construct a super-structure and infra-structure of main programs and subroutines for solving systems of ordinary differential equations by either i) directly using the codes `commondata.f90`, `rhs.f90`, `ode1.f90`, `ode2.f90`, `solveode.f90`, `runode`, along with the original code you write, `ode4.f90`, or ii) building your own super- and infra-structures. In either case, you should have a set of modular `Fortran` codes whose compilation and execution is coordinated by an executable script code.

3. Use your fourth order Runge-Kutta solver to estimate a solution to the problem considered in Section 22.3.1:
$$\frac{dy_1}{dt} = -y_1, \qquad y_1(0) = 1.$$
Take $\Delta t = 0.1$, estimate the solution for $t \in [0, 1]$, and report the error of your method at $t = 1$. Compare the error of the fourth order method to that of the first and second order methods at the same $t = 1$. *optional*: Demonstrate your method is a fourth order method by studying how the error converges as $\Delta t \to 0$.

4. Consider next the problem of Section 22.3.2 for a forced, damped Duffing equation:
$$\begin{aligned} \frac{dy_1}{dt} &= y_2, \qquad y_1(0) = 1, \\ \frac{dy_2}{dt} &= -\beta y_1 - \delta y_2 - \alpha y_1^3 + f \cos y_3, \qquad y_2(0) = 0, \\ \frac{dy_3}{dt} &= 1, \qquad y_3(0) = 0, \end{aligned}$$
with $\alpha = 1$, $\beta = -1$, $\delta = 0.22$, and $f = 0.3$. Using your fourth-order Runge-Kutta method, reproduce the results of Figs. 22.3 and 22.4.

Prepare your solution with the LaTeX text processor. The only code you need include is your new `ode4.f90`. As always, use at least one equation, use concise language, prepare beautiful figures. Because the codes may be a bit longer, take a *four page maximum.*