

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; a first order Runge-Kutta method is simply the forward Euler method. Let us consider the widely used *fourth order Runge-Kutta method*. Let us say we are solving N non-linear ordinary differential equations of the form

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

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 of Sec. 19.3.1:

$$\frac{dy_1}{dt} = -y_1, \quad y_1(0) = 1.$$

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 $\Delta t \rightarrow 0$.

Prepare your solution with the L^AT_EX 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. Take a *three page maximum*. Grading: technical merit, 50 points; æsthetics, 50 points.