Integration method

"A good ODE integrator should exert some adaptive control over its own progress, making frequent changes in its step size. Usually, the purpose of this adaptive step size control is to achieve some predetermined accuracy in the solution with minimum computational effort. Many small steps should tiptoe through treacherous terrain, while a few great strides should speed through the smooth uninteresting countryside. The resulting gains in efficiency are not mere tens of percents or factors of two; they can sometimes be factors of ten, a hundred, or more. Sometimes accuracy may be demanded not directly in the solution itself, but in some related conserved quantity that can be monitored." **- Numerical Recipes Chapter 16**

Numerical integration is a powerful tool that becomes fundamentally useful when we want to resolve equations that govern a system for which we don't know any analytical solution (apart of the steady state). This is even more true if the system that we study has few symmetry properties and, a lot of degrees of freedom or highly coupled variables like it's the case for KARMA's kinetics. To overcome this intricacy without having to assume tricky approximations, we have chosen to use a numerical integration method with an adaptive step size, and one of the best techniques of the kind is the "Adaptive 4^{th} order Runge-Kutta (A-RK4)". A 4^{th} order method, means that the total accumulated error is on the order of $O(h^4)$.

the Runge–Kutta methods are a family of implicit and explicit iterative methods used to numerically solve differential equation. The commonly used Euler method corresponds to the 1^{st} order RK method. Although, it is very easy to implement it remains limited in scope due to its sloppy behaviour and large truncation error.

Let's consider the following ODE with its initial condition :

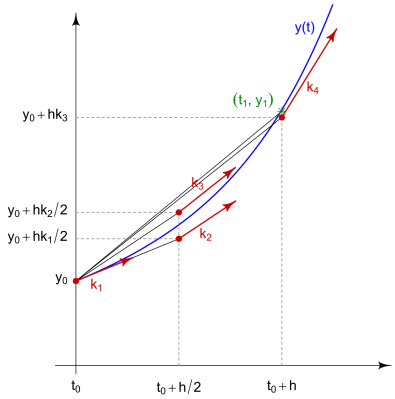
$$\frac{dy}{dt} = f(t, y), \qquad y(t_0) = y_0$$

The idea is that a value y_{n+1} (for $n \in N$) is approximated by the sum of the previous value y_n and the $h \times k$.

$$y_{n+1} = y_n + h \times k$$

Where $h = t_{n+1} - t_n$ is the time interval between the two values and k is the weighted average of the slope and is given for this method as:

$$k = \frac{k_1 + 2k_2 + 2k_3 + k_4}{6}$$



Where : $k_1 = hf(t_n, y_n)$ is the Euler slope calculated at point y_n.
$k_2 = hf(t_n + \frac{h}{2}, y_n + \frac{k_1}{2})$ the slop at $y_{n+\frac{1}{2}}$ calculated by using the Euler method (k_1).
$k_3 = h\ f(t_n + \frac{h}{2}, y_n + \frac{k_2}{2})$ the slop at the same point calculated via k_2.
And finally, $k_4 = hf(t_n + h, y_n + k_3)$ is the increment based on the slope at y_{n+1} via y_n and k_3.

Implementation of adaptive step size control requires that the stepping algorithm calculates its performance at each step, hence, its truncation error. Such information can be obtained by the calculating at each step two values for y_{n+1} ($y(1h)$ and $y(2h)$ in the next figure) one with an increment h and another one with an increment $2h$.

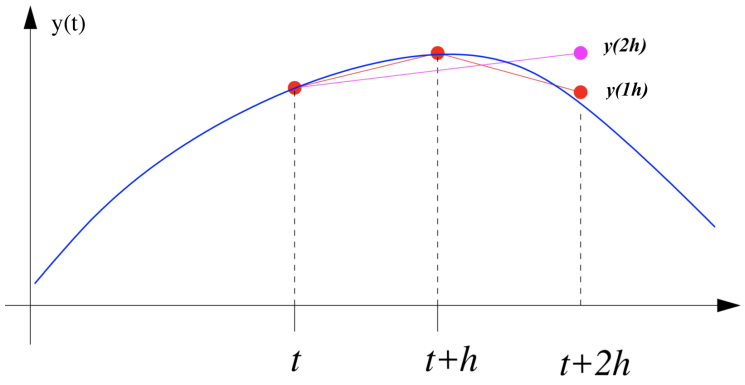
Given that RK4 gives exact solution to the 4^{th} order, we get :

$$y_{n+1} = y(1h) + 2ch^5 + O(h^6), \qquad c = const$$

$$y_{n+1} = y(2h) + c(2h)^5 + O(h^6)$$

The error is hence given by : $\epsilon \approx \frac{y(1h) - y(2h)}{30} \approx ch^5$

Therefore, in order to get a set precision δ, the tolerated error at each step should not exceed $h'\delta$. to include this criteria in the algorithm we simply need to impose the following conditions :



$$\begin{cases} \text{The calculator is too precise and the step size h should be reduced to gain calculation time.} & \text{if } h < h' \\ \text{The error exceeds } h'\delta\text{, and the step size should be reduced to gain precision} & \text{otherwise} \end{cases}$$

The optimal increment verifies the following formula:

$$\delta h' = |ch'^5| = |ch^5|(\frac{h'}{h})^5 = \frac{y(1h) - y(2h)}{30}(\frac{h'}{h})^5$$

Which finally yields :

$$h' = \frac{30\delta h^{\frac{5}{4}}}{|y(1h) - y(2h)|}$$