4) Stiff eq^{n}s:

→ These are linear differential eq^{n}s which if we try to solve by regular techniques like Euler's, RK, etc., the solution become unstable unless the value of step size is chosen to be smaller than a critical value.

- e.g. $y'(t) = -15\,y(t)$, $t \geq 0$, $y(0) = 1$

→ Such differential eq^{n}s quite often appear in problems involving rapidly decaying transient solutions.
e.g. Spring & damping system, Electrical circuits, etc.

→ Backward substitution can be used to solve such differential eq^{n}s.

→ ~~Numpy~~ Python function:

scipy.integrate.odeint

As per the documentation, this function can solve both stiff or non-stiff system of 1st order ode's.

14) Library : gsl_odeiv2.h

Solves n-D, 1[st] order systems:

$$\frac{dy_i(t)}{dt} = f_i(t, y_1(t), \ldots y_n(t))$$

Jacobian $J_{ij} = \frac{\partial f_i(t, y(t))}{\partial y_j}$

Defining ode: gsl_odeiv2_system

int (* function) (double t, const double y[], double dydt[],
void * params)

Define jacobian:

→ int (* jacobian) (double t, const double y[], double * dfdy,
double dfdt [], void * params)

Stepping functions: Advance solution from t to t+h.

gsl_odeiv2_step:

→ gsl_odeiv2_step * gsl_odeiv2_step_alloc (const gsl_odeiv2_step_type
* T, size_t dim)

↳ returns a pointer to newly allocated instance of stepping function of type T of dim dimensions.

→ ~~gsl~~ int gsl_odeiv2_step_reset (gsl_odeiv2_step * s)

↳ resets stepping function.

→ int gsl_odeiv2_step_apply (gsl_odeiv2_step * s, double t, double h,
double y[], double yerr[], const double dydt_in[], double dydt_out[],
const gsl_odeiv2_system * sys)

→ Applies stepping function s to the system of eq^n's sys, using step size h to advance from t → t+h.
New state is stored in y with abs. error yerr.

Algorithms:

→ gsl_odeiv2_step_rk2 : RK(2,3) - Explicit embedded.

→ gsl_odeiv2_step_rk4: Explicit RK4.

→ gsl_odeiv2_step_rkf45: Explicit Runge Kutta Fehlberg (45) method.

Any one of these can be used to define stepping function as following example:

e.g. gsl_odeiv2_step* gsl_odeiv2_step_alloc (const gsl_odeiv2_step_rk2 *T, size_t 10)

6^{th} order RK method:

Solve $\frac{dy}{dx} = f(x, y)$ subject to $y(x_0) = y_0$

Step-size : h. Error $\sim O(h^7)$

This can be done by making some evaluations at points bet^n ~~y_n & y_n~~ x_n & $x_n + h$.

$$y_{n+1} = y_n(x_n + h) + O(h^7)$$

$$y(x_n + h) = y(x_n) + h \left.\frac{dy}{dx}\right|_{x_n} + \frac{h^2}{2} \left.\frac{d^2y}{dx^2}\right|_{x_n}$$

$$+ \dots + \frac{h^6}{6!} \left.\frac{d^6 y}{dx^6}\right|_{x_n} + O(h^7)$$

Now $\frac{dy}{dx} = f(y, x)$

$$\frac{d^2y}{dx^2} = \frac{df}{dx} = \frac{\partial f}{\partial x} + \frac{\partial f}{\partial y}\frac{dy}{dx}$$

$$= f_x + f_y f \quad (f_x = \partial f / \partial x)$$

& similarly higher ~~od~~ order derivatives of y will be functions of f & partial derivatives of f wrt y & x.

Thus RHS $= y_n + h f(x_n, y_n) + \frac{h^2}{2}\left(f_x(x_n, y_n) + f_y f\right)$

$+ \dots$

LHS can be written as

$$y_{n+1} = y_n + h\left(\sum_{i=1}^{6} b_i K_i\right)$$

The solution to these system of equations is not unique. One of the solution is : (I referred to a Youtube video from Jacob Bishop's channel)

$$K_1 = f(x_n, y_n)$$

$$K_2 = f\left(x_n + \frac{h}{4}, y_n + \frac{K_1 h}{4}\right)$$

$$K_3 = f\left(x_n + \frac{h}{4}, y_n + \frac{k_1 h}{8} + \frac{k_2 h}{8}\right)$$

$$K_4 = f\left(x_n + \frac{h}{2}, y_n - \frac{k_2 h}{2} + k_3 h\right)$$

$$K_5 = f\left(x_n + \frac{3}{4}h, y_n + \frac{3}{16} K_1 h + \frac{9}{16} K_4 h\right)$$

$$K_6 = f\left(x_n + h, y_n - \frac{3}{7} k_1 h + \frac{2}{7} k_2 h + \frac{12}{7} k_3 h - \frac{12}{7} K_4 h + \frac{8}{5} K_5 h\right)$$

$$y_{n+1} = y_n + \frac{1}{90}\left(7K_1 + 32 k_3 + 12 k_4 + 32 k_5 + 7 K_6\right)$$

Notice that we require 6 function evaluations at each step.