Pruthvi
DHEP

4) Stiff differential equations is a differential equation for which certain numerical methods of solving ~~became~~ the equations are numerically unstable, unless the step-size is taken to be exteremely Small. A linear system, a system of differential equations is termed stiff if the ratio between the largest and the smallest eigen values is large.

- ~~Ex~~ Decay processes generally give stiff differential equations
- we use Backward integration methods to solve stiff differential equations.

- we can use scipy.integrate.solve-ivp - with method Set to 'BDF' for ~~stiff~~ solving stiff Differential ~~eq~~ equations. ~~[illegible] Backward~~ 'LASODA', 'Randu', ~~can~~ methods can also be used

14)
1) gsl-odeiv2-step-rk2 - Explicit Runge-kutta(2,3) method -

2) gsl-odeiv2-step-rk4 - 4th order classic Runge-kuttee method with Error estimation carried with step Doubling method

3) gsl_odeiv2_step_rkck - Explicit Runge-Kutta Cash-Karp (4,5) method.

- The Functions are used by first Difining ODE System.

gsl_odeiv2_system_sys = {,

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

, int (* jacobian) (double t, const double y[], double *dfdy, void *parms)

, size_t dimension

void *t prams

as we can solve multiple coupled DE's using y[]

Dimension of system of equatons -or number of equations

- Then we need to apply a driver ~~to the~~ which tells the method of stepping.

gsl_odeiv_driver *d = gsl_odeiv2_driver_alloc_y_new

(~~&sys~~ Def above) const gsl_odeiv2_system* sys,

step type to be used Defined above: ex gsl_odeiv2_step_rkck ← const, gsl, odeiv2_step type *T, const double hstart, constand douple epsabs, const double epsrel

hstart (as adaptive step size)

absolute and relative errors

Then we have to apply the driver

int gsl_odeiv2_driver_~~alloc~~ apply_~~fixed~~

(gsl_odeiv2_driver *d , double *t , const double t1,

double y[])

Define ranges of t [a, b]

(stores the solved values of y - should contain y_0 initially)

12) General Runge-Kutta method of order s can be written as

$$y_{t+h} = y_t + h \cdot \sum_{i=1}^{6} a_i k_i + O(h^{s+1})$$

where $k_i = h \sum_{j=1}^{6} B_{ij} f(k_j, t_n + \alpha_i h)$

~~we want to~~

from above we can see for 6th order f has to be evaluated 6 times. ~~Taking~~

$$y_{n+1} = y_n + \{9k_1 + 64k_3 + 49k_5 + 49k_6 + 9k_7\}/180$$

$k_1 = hf(x_n, y_n)$

$k_2 = hf(x_n + \nu h, y_n + \nu k_1)$

$k_3 = hf(x_n + h/2, y_n + \{(4\nu - 1)k_1 + k_2\}/(8\nu))$

$k_4 = hf(x_n + 2h/3, y_n + \{(10\nu - 2)k_1 + 2k_2 + 8\nu k_3\}/(27\nu))$

$k_5 = hf(x_n + (7 + \sqrt{21})h/14, y_n + \{[77\nu - 56] + [17\nu - 8]\sqrt{21})k_1$
$- 8(7 + \sqrt{21})k_2 + 48(7 + \sqrt{21})\nu k_3 - 3(21 + \sqrt{21})\nu k_4\}/392\nu$

$$k_6 = hf\left(x_n + (7-\sqrt{21})h/14,\ y_n + \{-5([287v-56] - [29v-8]\sqrt{21})k_1 - 40(7-\sqrt{21})k_2 + 320\sqrt{21}vk_3 + 3(21-121\sqrt{21})vk_4 + 396(6-\sqrt{21})vk_5\}/1960v\right)$$

$$k_7 = hf\left(x_n + h,\ y_n + \{15([30v-8] - [7v\sqrt{21}])k_1 + 120k_2 - 40(5+7\sqrt{21})vk_3 + 63(2+3\sqrt{21})vk_4 - 14(49-9\sqrt{21})vk_5 + 70(7+\sqrt{21})vk_6\}/(180v)\right)$$

- v may have any value other than ~~zero~~ zero.

Reference- An Explicit Sixth-Order Runge-Kutta Formula - ~~H.A. Luther~~
- H.A. Luther.

3) $y'' - 2y' + y = xe^x - x$

can be Divided as

$$y'' - y' - y' + y = xe^x - x$$

$$\frac{d}{dx}(y'-y) - (y'-y) = xe^x - x$$

taking $k = y' - y$

$$= k' = k + xe^x - x \quad (1)$$

$$, \quad y' = k + y \quad (2)$$

are a pair of DE that have been solved with inital value of y as 0 and k as 0 (as $y, y' = 0$)