

ALGORITHM:

4TH ORDER RUNGE-KUTTA FOR SYSTEMS OF EQUATIONS

A second order differential equation can be written as a system of two first order differential equations. For example, if the differential equation to be solved defines the acceleration of a particle, then integrating once yields the particle's velocity, while integrating a second time yields the particle's position. Suppose that the equation of acceleration is

$$\frac{d^2x}{dt^2} = 2xt^3$$

This second order differential equation can be written as a system of two first order equations:

$$\begin{aligned}\frac{dx}{dt} &= v \\ \frac{dv}{dt} &= 2xt^3\end{aligned}$$

where v is the velocity of the particle. Thus instead of one second order equation we have a system of two first order equations. A *state vector* can be defined which contains two elements, the position and velocity of the particle at some time t . This state vector is

$$\begin{bmatrix} x \\ v \end{bmatrix}$$

Taking the derivative of this state vector with respect to time yields

$$\frac{d}{dt} \begin{bmatrix} x \\ v \end{bmatrix} = \begin{bmatrix} v \\ 2xt^3 \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} x' \\ v' \end{bmatrix} = \begin{bmatrix} v \\ 2xt^3 \end{bmatrix}$$

In the previous Runge-Kutta programming assignment, your program probably contained a separate function which evaluated the derivative. In that function, you had a statement similar to

```
xpertime = -10.d0 * t
```

where `xpertime` is the derivative given by the first order differential equation. For systems of equations, our function which evaluates the derivative must contain the differential equation for each element in the state vector. If we call our state vector `state` and its derivative `stateprime`, then each of these variables will be a 2-element array (vector). We can define the first element of the state vector to be position and the second element velocity. If we do this, then the code statements appearing in the function to evaluate the derivative would look similar to:

```
stateprime(1) = state(2)
stateprime(2) = 2.d0 * state(1) * t**t
```

Note that `state(1)` is used instead of `x` to calculate `stateprime(2)`. This is because the particle's position is the first element of the state vector, and the variable `x` will not appear in the program.

The algorithm to accomplish integration of systems is below.

Purpose: to numerically solve a system of first-order differential equations.

Requires: an external function to calculate the derivatives, which is called `getdx` for this example.

Inputs: current state vector `state`; dimension of state vector `dim`; time t ; increment `delta_t`

Output: new state vector `new_state`.

Step 1: Define an array `offset` = [0, 0.5, 0.5, 1.0]

Step 2: Compute the K values for the Runge-Kutta step

For $ii = 1$ to 4, Do the following

```
    if ii = 1 then
        set x1 = state
    else
        set x1 = state + offset(ii) * K(ii-1) for each element in the state vector
    endif
```

Set $t1 = t + offset(ii) * delta_t$

$delta_x = getdx(x1)$ (Compute derivatives based on current $x1$)

Calculate K values for each element in the state vector:

$K(ii,j) = delta_x(j) * delta_t$ for all $jj = 1$ to dim

End For Loop

Now, update each element in the state vector:

For $jj = 1$ to dim

$xx = K(1,jj) + 2*K(2,jj) + 2*K(3,jj) + K(4,jj)$

$new_state = state + xx/6$

End For Loop