numerically solving the kepler problem

Nikki Rider

2022

np.arange(4) → [0 1 2 3]

the y-array you are given as an input parameter in your function get_dydx:

$$y = \begin{array}{c} \\ 2\,\text{nbodies} \\ \\ \end{array} \left[\begin{array}{c} x_0 \\ y_0 \\ x_1 \\ y_1 \\ \vdots \\ x_{N-1} \\ y_{N-1} \\ vx_0 \\ vy_0 \\ vx_1 \\ vy_1 \\ \vdots \\ vx_{N-1} \\ vy_{N-1} \end{array}\right] \begin{array}{c} \boxed{0} \\ 1 \\ \boxed{2} \\ 3 \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{array}$$

the x_i values have even indices starting with 0 and ending with N−1 (nbodies−1 in code)

we can use this pattern to write an array of the indices where we can access the x_i values:

[0 1 2 3 ⋯ nbodies-1]

indx = np.arange(nbodies) * 2

gives [0 2 4 ⋯ 2*nbodies-1]

look up the documentation for the np.arange function, and write index arrays for the y, vx, and vy variables indy = np.arange(nbodies)*2 + 1

→ [1 3 5 ...... nbodies]

y[indy] → [y_0 y_1 y_2 ......]

indvx = np.arange(nbodies)*2 + 2*nbodies

indvy = indvx + 1

y[indvx] → [VX_0 VX_1 VX_2 ......]

y[indx] gives $[x_0 \; x_1 \; x_2 \; \cdots \; x_N]$

y[indx[2]] gives x_2

using this method, you can make arrays for all x, y, u, and v for all planets

$\dot{x}_i = vx_i$ ← $\overbrace{\text{dydx[indx]}}^{\dot{x}} = \text{y[indvx]}$ (vx)

where i refers to one body and j is summed over all other bodies

y[indx[j]]

$\dot{y}_i = vy_i$ ←

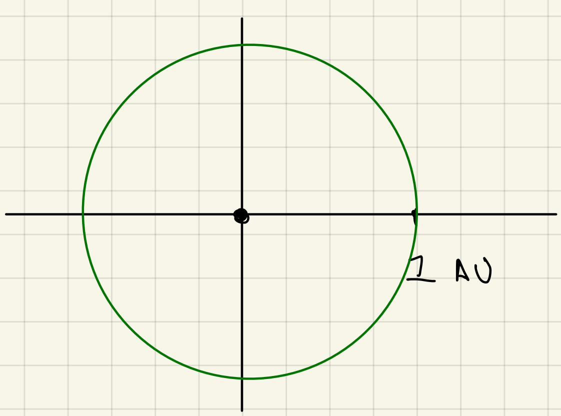
$$\boxed{\begin{aligned} \dot{vx}_i &= G \sum_j m_j \frac{(x_j - x_i)}{R_{ji}^3} \\ \dot{vy}_i &= G \sum_j m_j \frac{(y_j - y_i)}{R_{ji}^3} \end{aligned}}$$

where $R_{ji} \equiv |\vec{r}_j - \vec{r}_i| = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$

```
for i in range(nb)
    gravx = 0.0 ;
    gravy = 0.0 ;
    for j ______
        if i != j :
            gravx += ______

    dydx[indvx[i]] = gravx
```

$\dot{vx}_i =$



use nested for loops to calculate $\dot{u}_i$ and $\dot{v}_i$ for each planet, summing the forces due to the others

return dydx as a numpy array in the same order as y

$$dydx = \begin{bmatrix} \dot{x}_0 \\ \dot{y}_0 \\ \dot{x}_1 \\ \dot{y}_1 \\ \vdots \\ \dot{x}_N \\ \dot{y}_N \\ \dot{u}_0 \\ \dot{v}_0 \\ \dot{u}_1 \\ \dot{v}_1 \\ \vdots \\ \dot{u}_N \\ \dot{v}_N \end{bmatrix}$$