Life as a Trisolarian

Adaptive Solving of ODEs

Sarah Draves

Original Project Concept

- Based on the problems of the alien world Trisolaris in the book The Three-Body Problem
- Simulate motion of three stars and one massless planet in 3D with high resolution using an adaptive ODE solver for the classical gravity equation

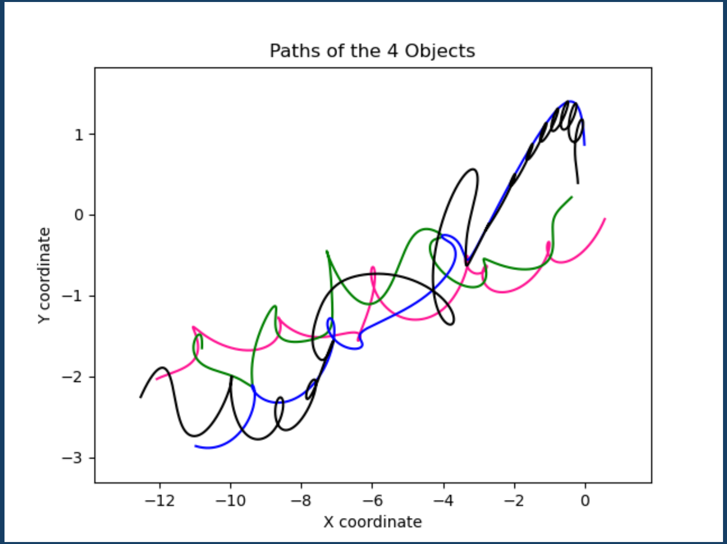
$$m_i \frac{d^2 \mathbf{q}_i}{dt^2} = \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{G m_i m_j \left(\mathbf{q}_j - \mathbf{q}_i\right)}{\left\|\mathbf{q}_j - \mathbf{q}_i\right\|^3} = -\frac{\partial U}{\partial \mathbf{q}_i}$$

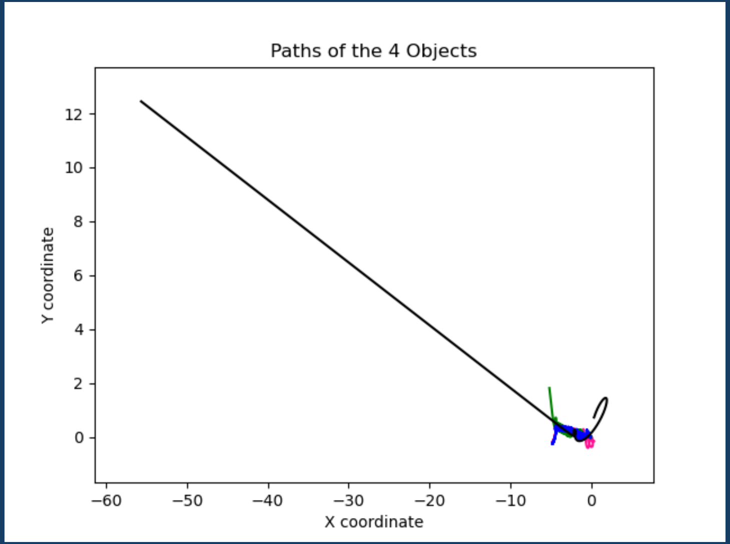
- Look at the system from the point of view of the planet and predict what will happen to the planet with measurement error introduced

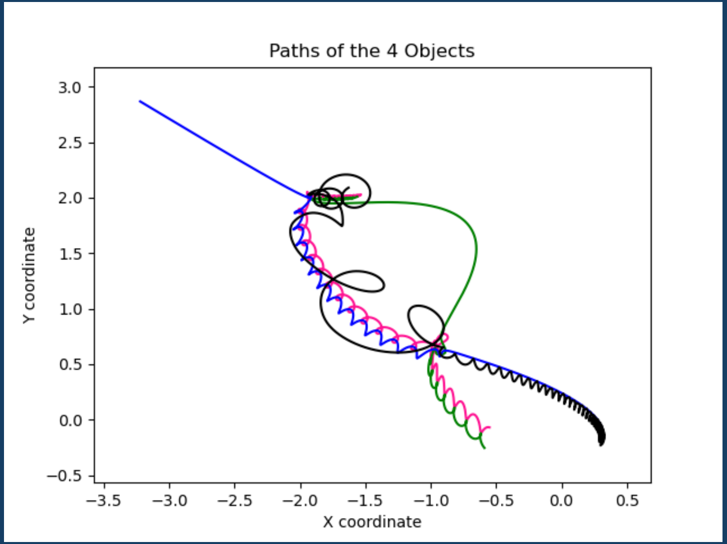
Final Project Concept

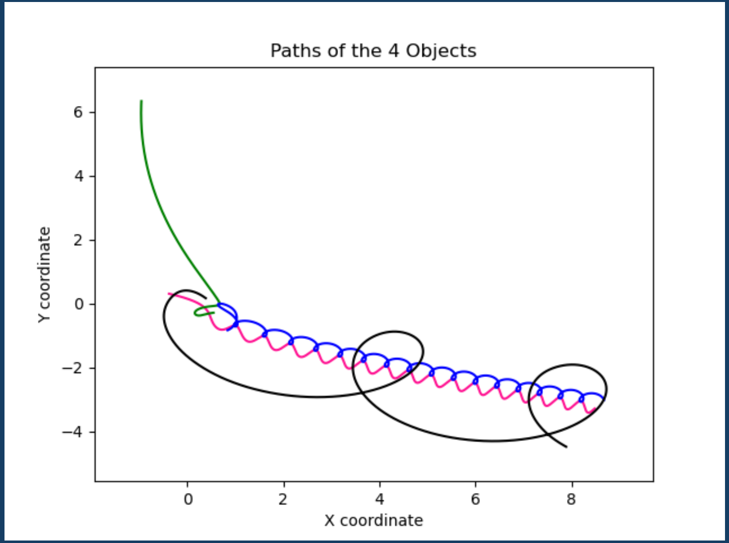
- Created my own adaptive RK4 integrator that visually matches Scipy’s RK45 method when the error tolerance is set the same and runs in a reasonable amount of time
- Plotted and animated different scenarios in 2D with different numbers of stars and planets
- Made interactive program where the user can enter their own initial conditions for different scenarios and see it plotted or animated
- Ran out of time for looking at the system from the planet perspective

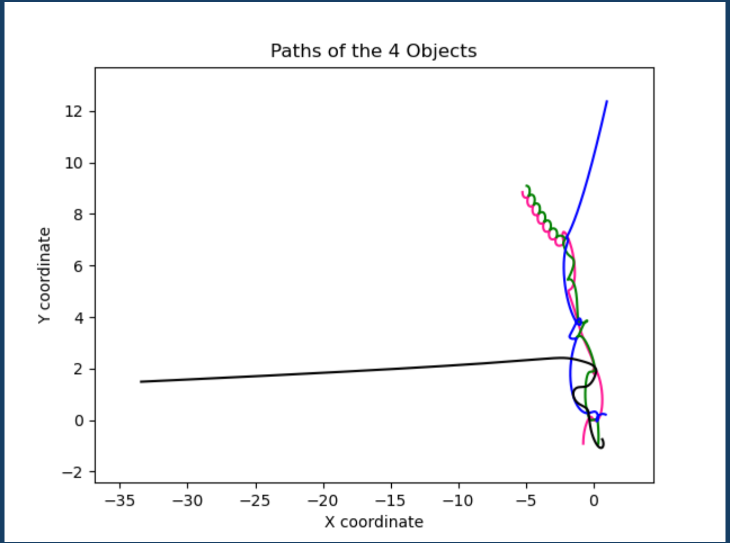
The Planet Suffers

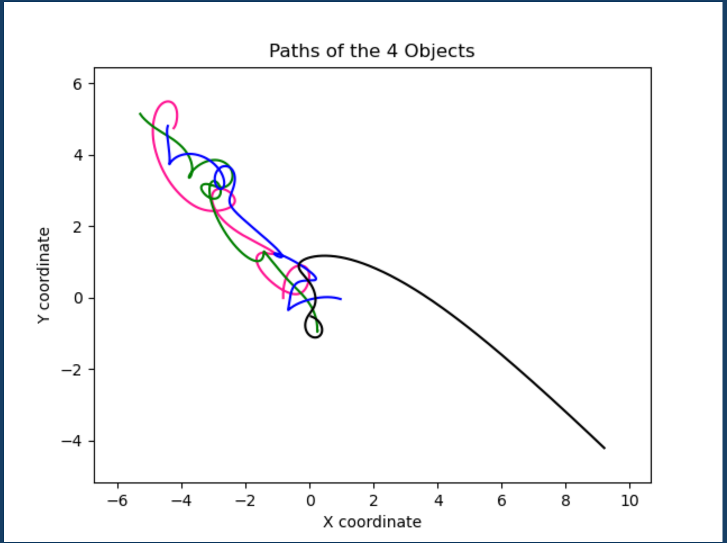




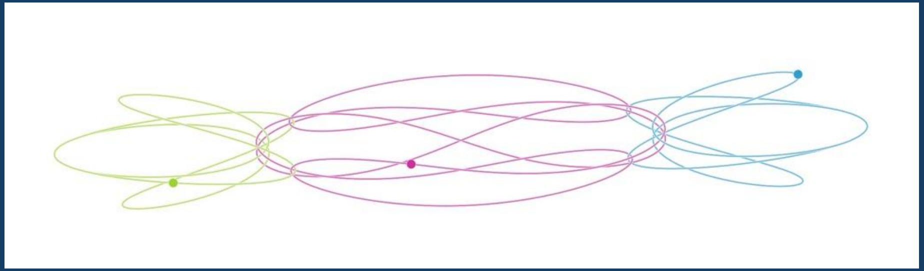


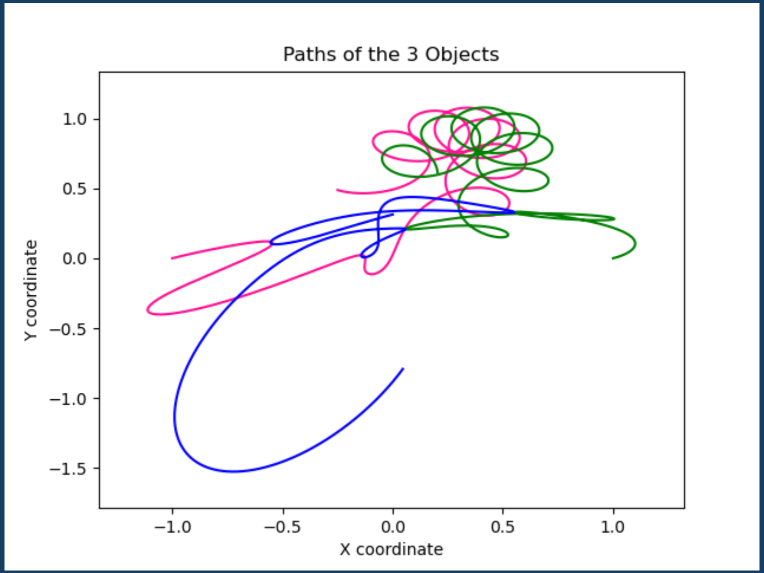




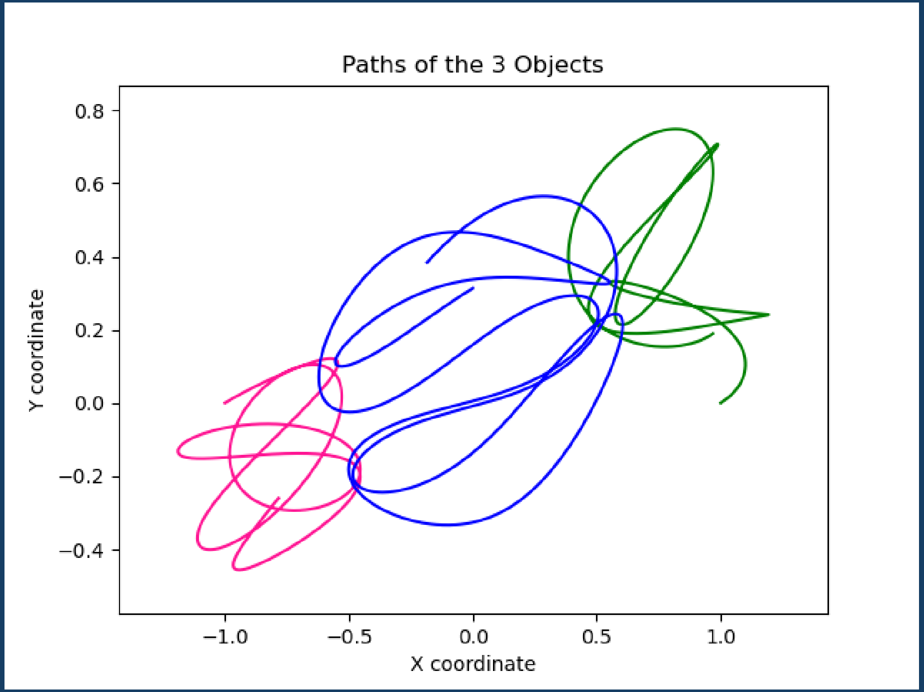


Adaptive Integration

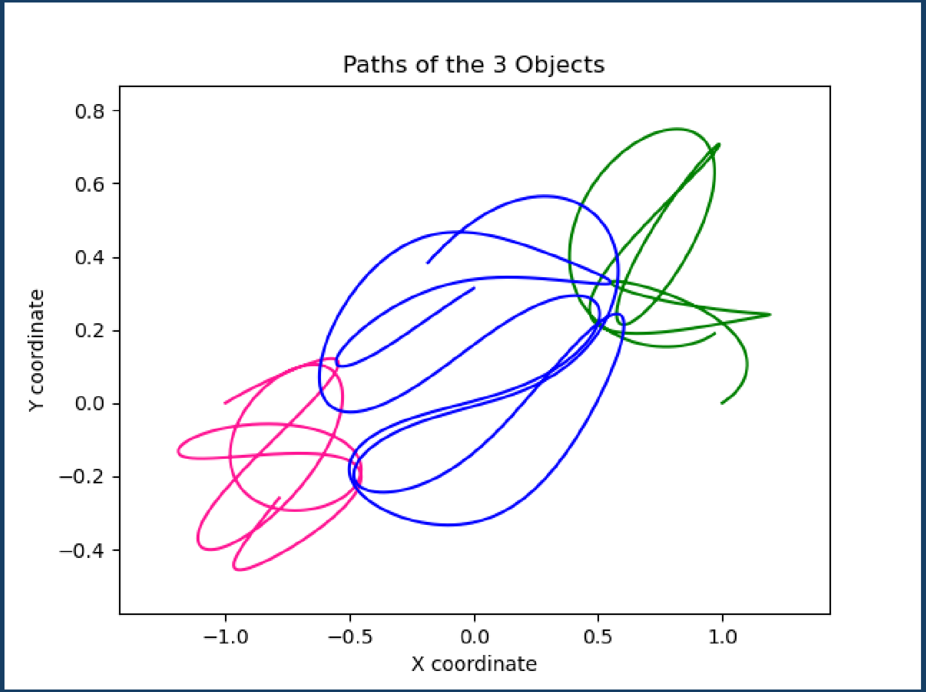

From https://observablehq.com/@rreusser/periodic-planar-three-body-orbits



Solver: My RK4, Step size: 0.00001, Time to run: 25.4231 seconds

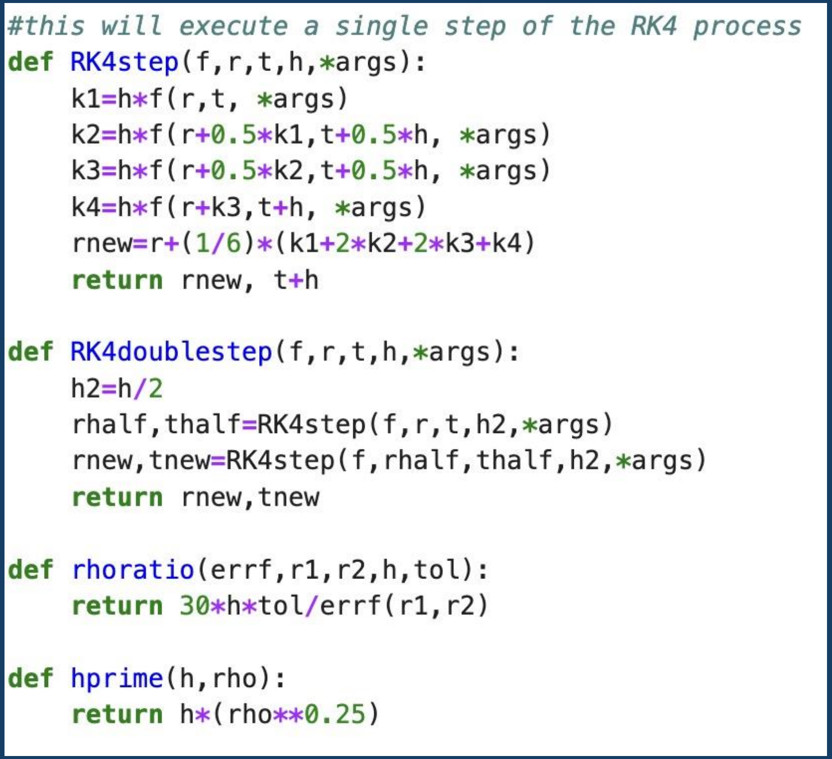


Solver: Scipy's RK45, Tolerance: 1e-12, Time to run: 0.4417 seconds

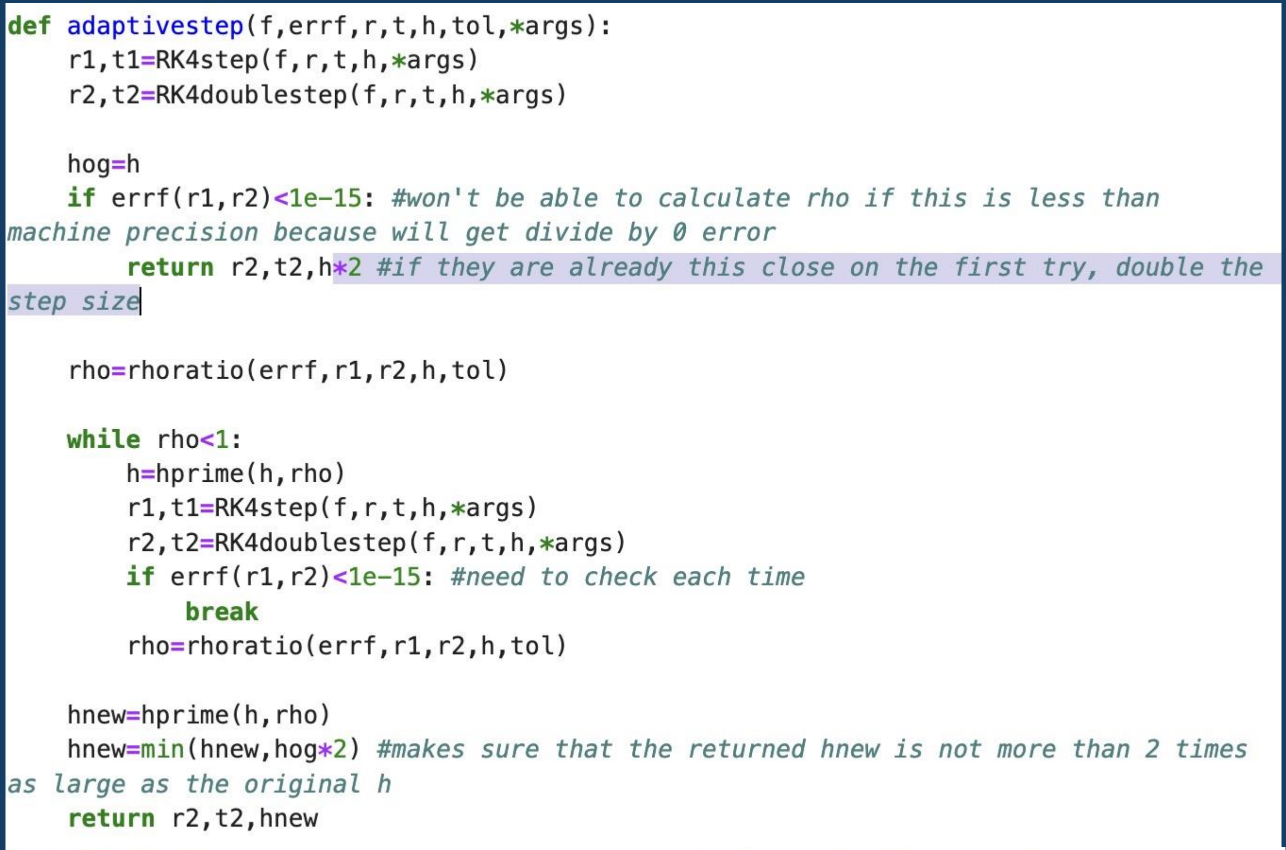


Solver: My Adaptive RK4, Tolerance: 1e-12, Time to run: 4.3570 seconds

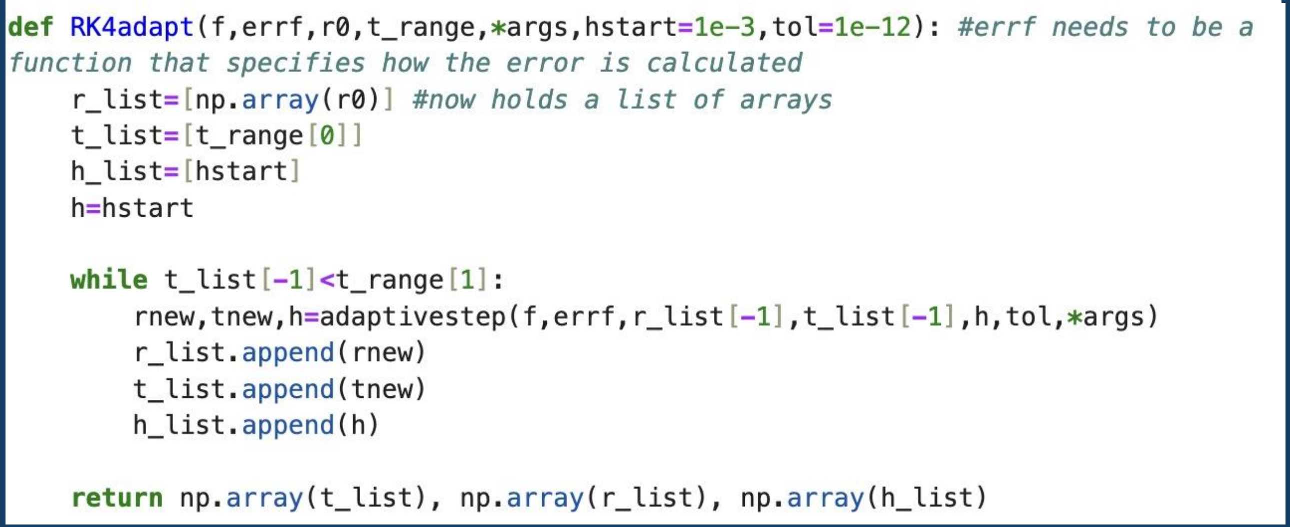
Adaptive Integration Method



```python
#this will execute a single step of the RK4 process
def RK4step(f,r,t,h,*args):
    k1=h*f(r,t, *args)
    k2=h*f(r+0.5*k1,t+0.5*h, *args)
    k3=h*f(r+0.5*k2,t+0.5*h, *args)
    k4=h*f(r+k3,t+h, *args)
    rnew=r+(1/6)*(k1+2*k2+2*k3+k4)
    return rnew, t+h

def RK4doublestep(f,r,t,h,*args):
    h2=h/2
    rhalf,thalf=RK4step(f,r,t,h2,*args)
    rnew,tnew=RK4step(f,rhalf,thalf,h2,*args)
    return rnew,tnew

def rhoratio(errf,r1,r2,h,tol):
    return 30*h*tol/errf(r1,r2)

def hprime(h,rho):
    return h*(rho**0.25)
```

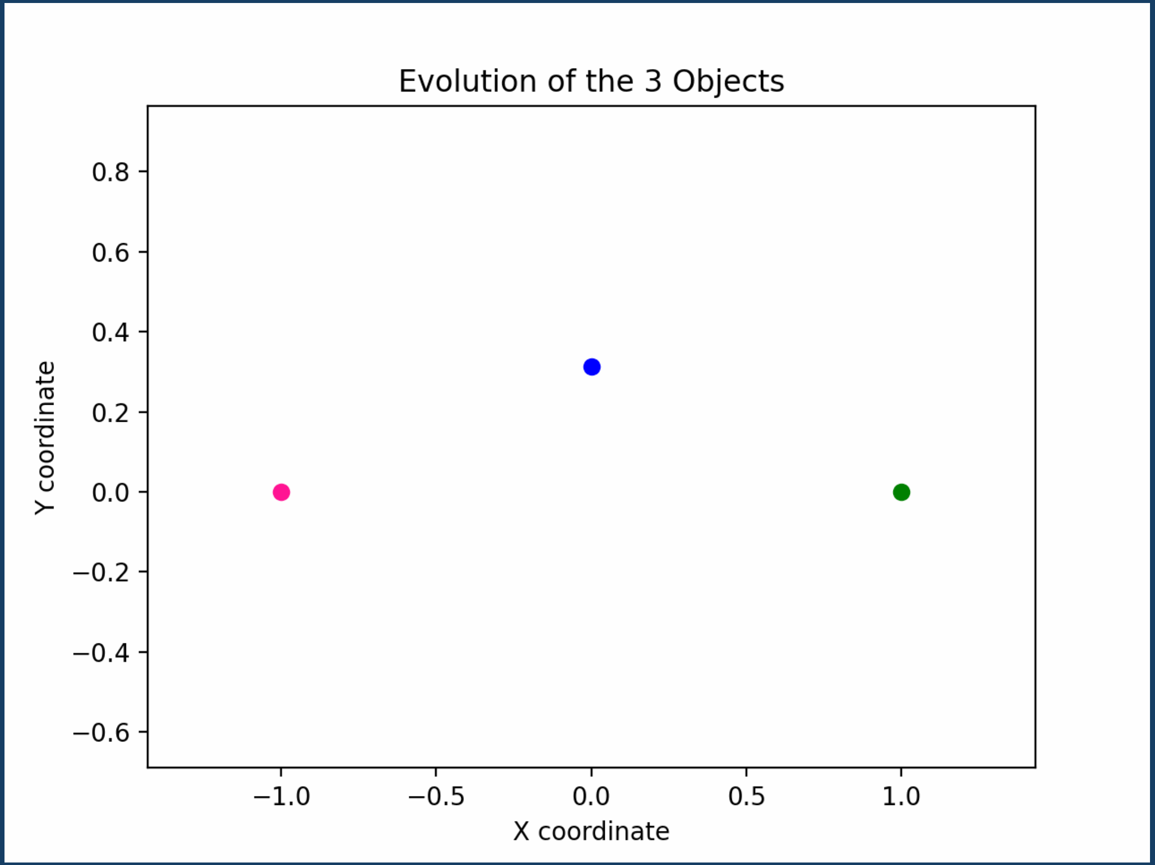


```python
def adaptivestep(f,errf,r,t,h,tol,*args):
    r1,t1=RK4step(f,r,t,h,*args)
    r2,t2=RK4doublestep(f,r,t,h,*args)

    hog=h
    if errf(r1,r2)<1e-15: #won't be able to calculate rho if this is less than
machine precision because will get divide by 0 error
        return r2,t2,h*2 #if they are already this close on the first try, double the
step size

    rho=rhoratio(errf,r1,r2,h,tol)

    while rho<1:
        h=hprime(h,rho)
        r1,t1=RK4step(f,r,t,h,*args)
        r2,t2=RK4doublestep(f,r,t,h,*args)
        if errf(r1,r2)<1e-15: #need to check each time
            break
        rho=rhoratio(errf,r1,r2,h,tol)

    hnew=hprime(h,rho)
    hnew=min(hnew,hog*2) #makes sure that the returned hnew is not more than 2 times
as large as the original h
    return r2,t2,hnew
```

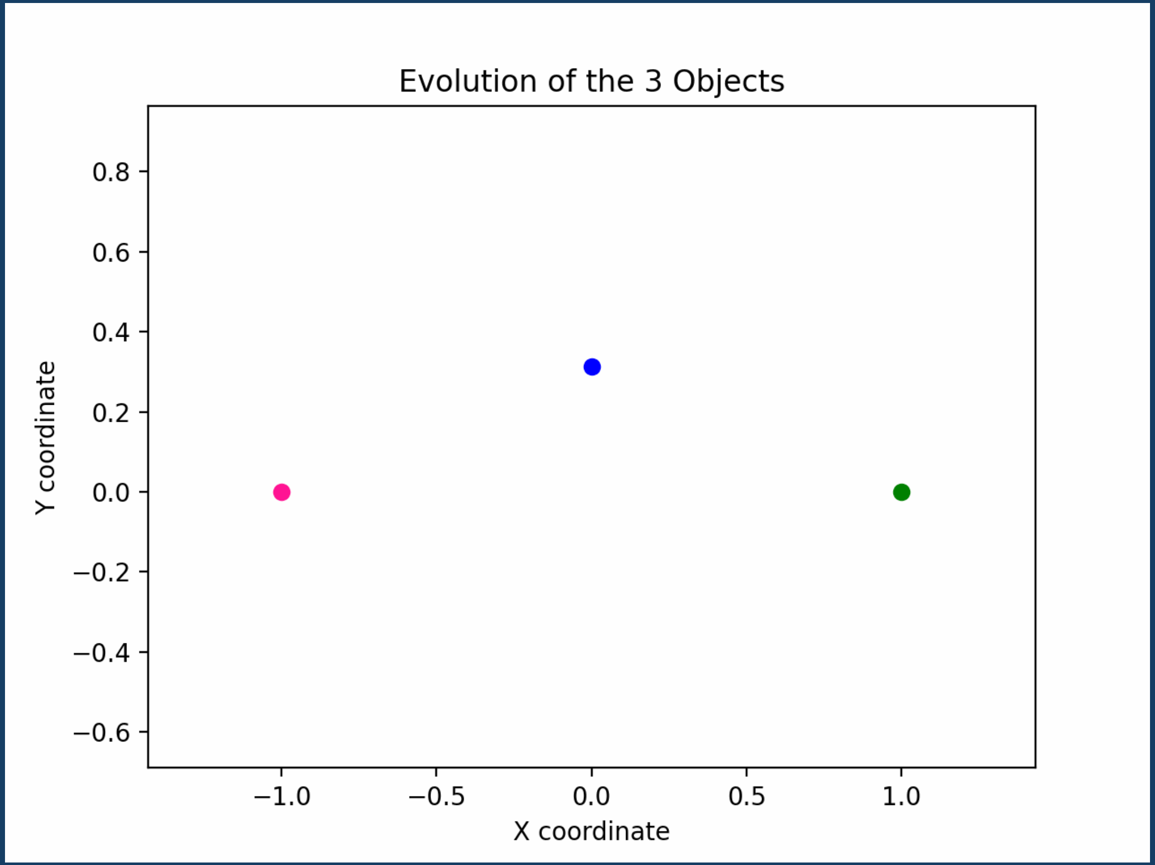


```python
def RK4adapt(f,errf,r0,t_range,*args,hstart=1e-3,tol=1e-12): #errf needs to be a
function that specifies how the error is calculated
    r_list=[np.array(r0)] #now holds a list of arrays
    t_list=[t_range[0]]
    h_list=[hstart]
    h=hstart

    while t_list[-1]<t_range[1]:
        rnew,tnew,h=adaptivestep(f,errf,r_list[-1],t_list[-1],h,tol,*args)
        r_list.append(rnew)
        t_list.append(tnew)
        h_list.append(h)

    return np.array(t_list), np.array(r_list), np.array(h_list)
```

Adaptive Integration Animation Issue

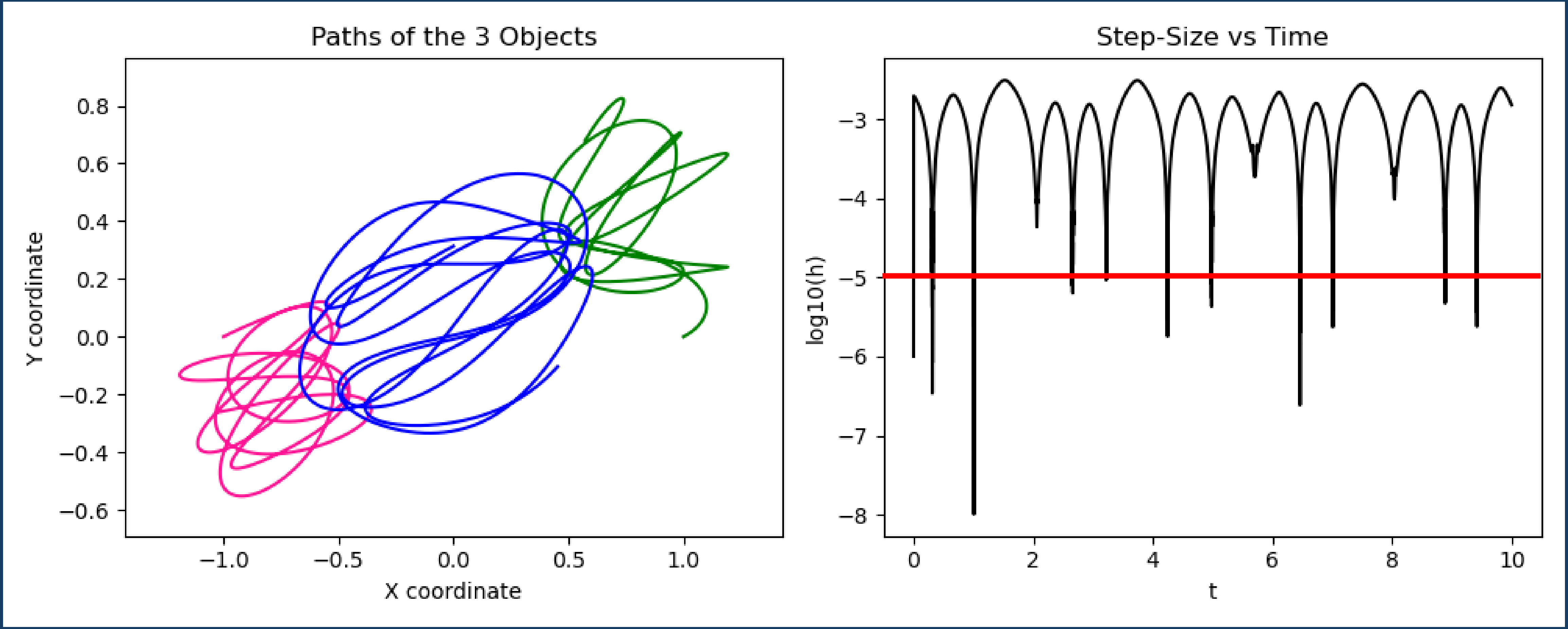


Animating raw output from my adaptive integrator



Animating after resampling 10,000 points that are evenly spaced in time using scipy's cubic interpolation method

Adaptive Integration H vs T



Live Demo

Sources

- The Three-Body Problem by Liu Cixin and 3 Body Problem from Netflix for inspiration
- https://en.wikipedia.org/wiki/Three-body_problem for background on periodic orbits
- https://observablehq.com/@rreusser/periodic-planar-three-body-orbits for initial conditions of periodic orbits
- Scipy documentation