Homework 1

Homework 1-1

0.0/1.0 point (graded)

Modify the original code example used in Part 3, which used the Euler method, to solve Eq.(A22) with a 2nd order Runge-Kutta method. This can be achieved by replacing the following code fragment with the corresponding RK code.

```
for i in range(step-1):
    y[i+1]=y[i]-dt*y[i]
```

Choose the most appropriate solution from the code fragment shown below (C11, C12, C13, C14).

```
# C11
y1 = np.zeros(step)
for i in range(step-1):
    y1[i]=y[i]-dt*y[i]
    y[i+1]=y[i]-0.5*dt*y1[i]
```

```
# C12
y1 = np.zeros(step)
for i in range(step-1):
    y1[i]=y[i]-0.5*dt*y[i]
    y[i+1]=y[i]-dt*y1[i]
```

```
# C13
y1 = np.zeros(step)
y2 = np.zeros(step)
y3 = np.zeros(step)
for i in range(step-1):
    y1[i]=y[i]-0.5*dt*y[i]
    y2[i]=y[i]-0.5*dt*y1[i]
    y3[i]=y[i]-dt*y2[i]
    y[i+1]=y[i]-dt*(y[i]+y1[i]+y2[i]+y3[i])/4.0
```

```
# C14
y1 = np.zeros(step)
y2 = np.zeros(step)
y3 = np.zeros(step)
for i in range(step-1):
    y1[i]=y[i]-0.5*dt*y[i]
    y2[i]=y[i]-0.5*dt*y1[i]
    y3[i]=y[i]-dt*y2[i]
    y[i+1]=y[i]-dt*(y[i]+2.0*y1[i]+2.0*y2[i]+y3[i])/6.0
```

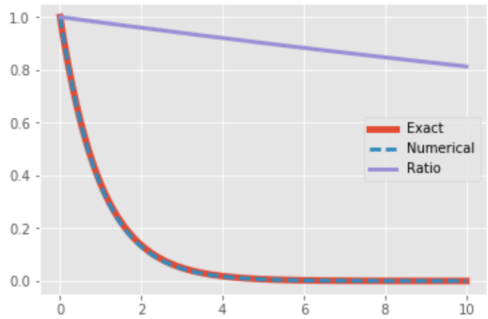
Submit You have used 0 of 2 attempts

Homework 1-2

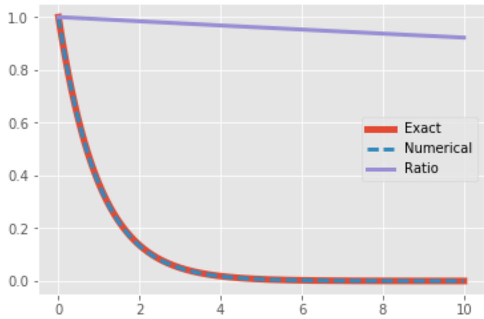
0.0/2.0 points (graded)

Perform the simulation using the modified code for the previous homework with the 2nd order Runge-Kutta method, and make the same graph we previously made using the Euler method in Part 3. Which of the following graphs (G21, G22, G23, G24) is the closest to what you obtained?

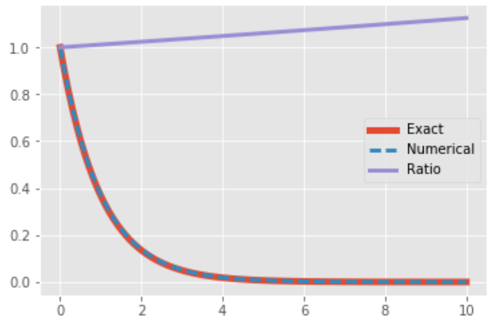
G21



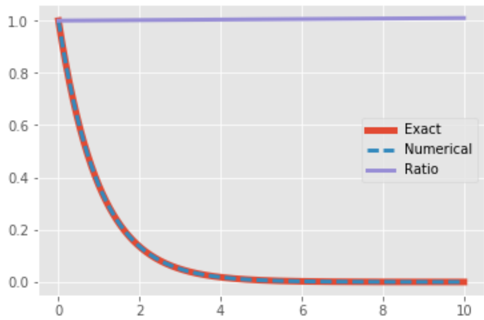
G22



G23



G24



- G21
- G22
- G23
- G24

Submit

You have used 0 of 2 attempts

Homework 1-3

0.0/1.0 point (graded)

Modify the original code used in Part 4, which used the Euler method to solve for the motion of the harmonic oscillator, with a 4th order Runge-Kutta method. This can be achieved through a suitable modification of the following *animate* function.

```
def animate(i):
    global R,V,F,Rs,Vs,time,Et
    V = V*(1-zeta/m*dt)-k/m*dt*R
    R = R + V*dt
    Rs[0:dim,i]=R
    Vs[0:dim,i]=V
    time[i]=i*dt
    Et[i]=0.5*m*np.linalg.norm(V)**2+0.5*k*np.linalg.norm(R)**2
    particles.set_data(R[0], R[1])
    line.set_data(Rs[0,0:i], Rs[1,0:i])
    title.set_text(r"$t = {0:.2f},E_T = {1:.3f}$".format(i*dt,Et[i]))
    return particles,line,title
```

Choose the most appropriate animate function from the codes shown below (C21, C22, C23, C24).

```
# C21
R1    = np.zeros(dim)
V1    = np.zeros(dim)
def animate(i):
    global R,V,F,Rs,Vs,time,Et
    V1 = V - zeta/m*dt*V - k/m*dt*R
    R1 = R + V*dt
    V = V - V1*zeta/m*0.5*dt - k/m*0.5*dt*R1
    R = R + V1*dt
    Rs[0:dim,i]=R
    Vs[0:dim,i]=V
    time[i]=i*dt
    Et[i]=0.5*m*np.linalg.norm(V)**2+0.5*k*np.linalg.norm(R)**2
    particles.set_data(R[0], R[1])
    line.set_data(Rs[0,0:i], Rs[1,0:i])
    title.set_text(r"$t = {0:.2f},E_T = {1:.3f}$".format(i*dt,Et[i]))
    return particles,line,title
```

```
# C22
R1    = np.zeros(dim)
V1    = np.zeros(dim)
def animate(i):
    global R,V,F,Rs,Vs,time,Et
    V1 = V - zeta/m*0.5*dt*V - k/m*0.5*dt*R
    R1 = R + V*0.5*dt
    V = V - V1*zeta/m*dt - k/m*dt*R1
    R = R + V1*dt
    Rs[0:dim,i]=R
    Vs[0:dim,i]=V
    time[i]=i*dt
    Et[i]=0.5*m*np.linalg.norm(V)**2+0.5*k*np.linalg.norm(R)**2
    particles.set_data(R[0], R[1])
    line.set_data(Rs[0,0:i], Rs[1,0:i])
    title.set_text(r"$t = {0:.2f},E_T = {1:.3f}$".format(i*dt,Et[i]))
    return particles,line,title
```

```
# C23
R1    = np.zeros(dim)
V1    = np.zeros(dim)
```

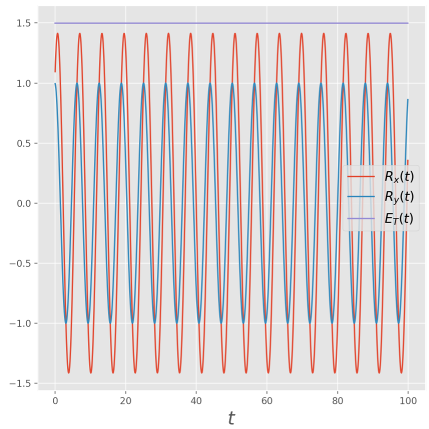
Submit You have used 0 of 2 attempts

Homework 1-4

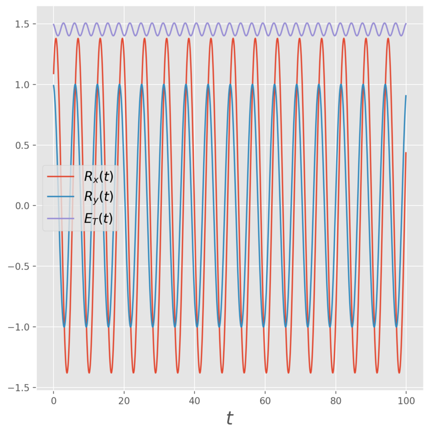
0.0/2.0 points (graded)

Set the friction constant to 0 ($\zeta = 0.0$), perform the simulation using the modified code for the previous homework with the 4th order Runge-Kutta method, and plot $R_x(t)$ vs $R_y(t)$. Which of the following graphs (G31, G32, G33, G34) is the closest to what you obtained?

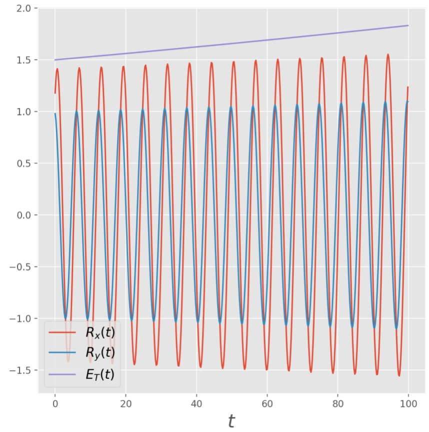
G31



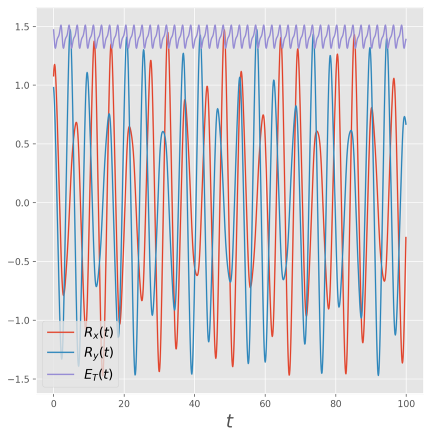
G32



G33



G34



- G31
- G32
- G33
- G34

Submit You have used 0 of 2 attempts