# Equations of Motion

Computational Physics

Orbital Motion

# Outline

- Fourth Order Runge-Kutta Method
- Equation of motion in 3 dimensions
- Projectile Motion Problem
- Orbit Equations

# Second Order Runge-Kutta

$$\frac{dy}{dt} = f(y,t)$$

**Differential Equation**

$$y(t+\frac{\Delta t}{2}) = y(t) + f(y(t),t)\frac{\Delta t}{2}$$

**Estimate value of y at half-step (Euler Method)**

$$y(t+\Delta t) = y(t) + f(y(t+\frac{\Delta t}{2}),t+\frac{\Delta t}{2})\Delta t$$

**Use value at half-step to find new estimate of derivative**

# Fourth Order Runge-Kutta

$$y(t+\Delta t) = y(t) + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)\Delta t$$

Estimate of derivative in interval

$$k_1 = f(y(t), t)$$

Value at beginning of interval

$$k_2 = f(y(t) + k_1\frac{\Delta t}{2}, t + \frac{\Delta t}{2})$$

$$k_3 = f(y(t) + k_2\frac{\Delta t}{2}, t + \frac{\Delta t}{2})$$

Two estimates of value at mid-point

$$k_4 = f(y(t) + k_3\Delta t, t + \Delta t)$$

Estimate of value at end of interval

# Motion in Three Dimensions

Independent Equations for each dimension

$$\frac{d^2x}{dt^2} = \frac{F_x}{m}$$

$$\frac{d^2y}{dt^2} = \frac{F_y}{m}$$

$$\frac{d^2z}{dt^2} = \frac{F_z}{m}$$

Single Vector Equation

$$\frac{d^2\ \vec{r}}{dt^2} = \frac{\vec{F}}{m}$$

in Python we write vectors:

$r = [\, x\,,\ y\,,\ z\,]$

$F = [\, F_x, F_y, F_z\,]$

# 3D Solution in Python
## *Independent Equations - Euler Method*

Initialization

```
X[0] = 0.
Y[0] = 0.
Z[0] = 0.
VX[0] = 0.
VY[0] = 0.
VZ[0] = 0.
# FX, FY, FZ are components of force
for i in range(n):
    VX[i+i] = VX[i] + FX[i]/m*dt
    VY[i+i] = VY[i] + FY[i]/m*dt
    VZ[i+i] = VZ[i] + FZ[i]/m*dt

    X[i+i] = X[i] + VX[i]*dt
    Y[i+i] = Y[i] + VY[i]*dt
    Z[i+i] = Z[i] + VZ[i]*dt
```

Velocity Components

Position Components

# 3D Solution in Python

## *Vector Equations - Euler Method*

**Initialization**

**Velocity Vector**

**Position Vector**

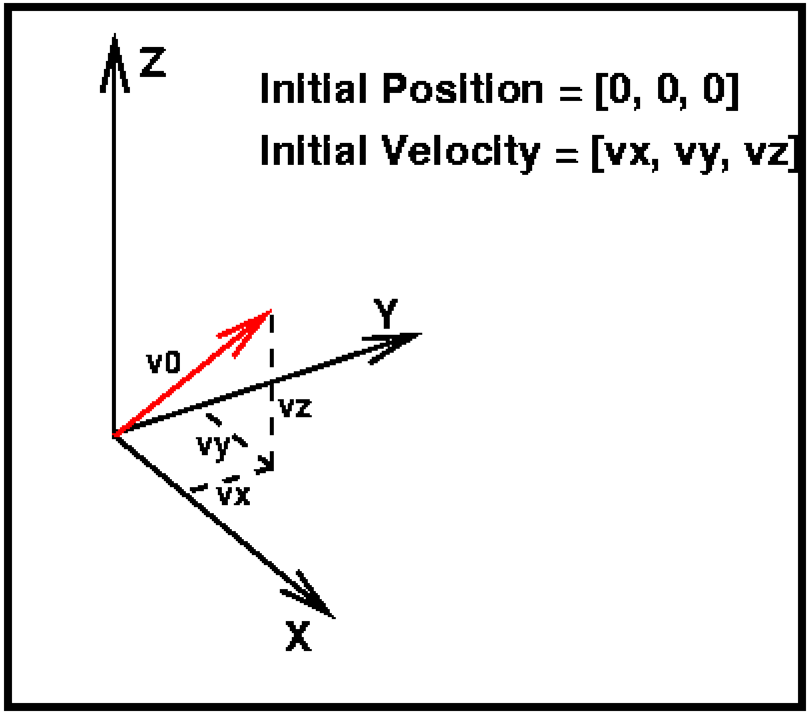
```
X0 = [0.,0.,0.]
V0 = [0.,0.,0.]

X = np.zeros( (nsteps,3) )
V = np.zeros( (nsteps,3) )
# use colon operator to set vectors
X[0,:] = X0
V[0,:] = V0

# F[i,:] has FX, FY, FZ] for each step i
for i in range(nsteps):
    V[i+1,:] = V[i,:] +  F[i,:]/m*dt
    X[i+1,:] = X[i,:] + V[i,:]*dt
```

# Projectile Motion Problem

- Motion of particle under gravity, and eventually other realistic forces.
- Initial Conditions:
  - specify location of beginning of trajectory
  - specify initial velocity



# Equation of Motion

## *Gravity Only*

$$\frac{d^2x}{dt^2} = \frac{F_x}{m} = 0$$

$$\frac{d^2y}{dt^2} = \frac{F_y}{m} = 0$$

$$\frac{d^2z}{dt^2} = \frac{F_z}{m} = -g$$

$$\frac{d^2\,\vec{r}}{dt^2} = \frac{\vec{F}}{m} = -g\,\hat{z}$$

**Gravity is only force:**
**Acceleration in -z direction**

# Constants of the Motion
## *Gravity Only*

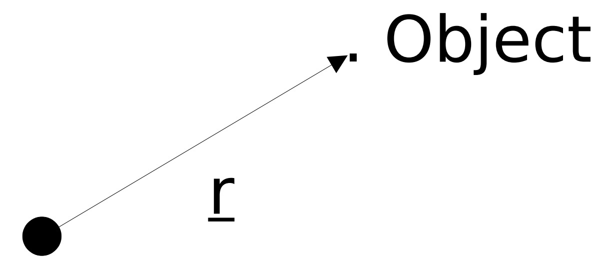
- Constants of motion are useful for evaluating whether your program works!
- No Force in X and Y directions:
  - momentum in X and Y conserved
- Force of gravity depends on position only
  - total energy is conserved
  - potential energy = m g z
  - kinetic energy = $\frac{1}{2} m |v|^2$
  - total energy: $E = \frac{1}{2} m |v|^2 + m g z$

# Orbit Problem
# Equation of Motion

- Second Order ODE
- Radial Force dependent on position only:
  - Angular Momentum conserved; Motion in a plane.
  - Energy conserved.

$$\frac{d^2 \vec{r}}{dt^2} = \frac{\vec{F}}{m}$$

$$\vec{F} = -\frac{G\, M_{\odot}\, m}{r^2}\,\hat{r}$$



# The r unit vector.

- Gravitational Force is radial, so need unit vector in r direction to derive force.
- A convenient way to look at this, for python programs is:

$$\hat{r} = \frac{\vec{r}}{|\vec{r}|}$$

- Magnitude of r is numpy.linalg.norm(r)

# Initial Conditions

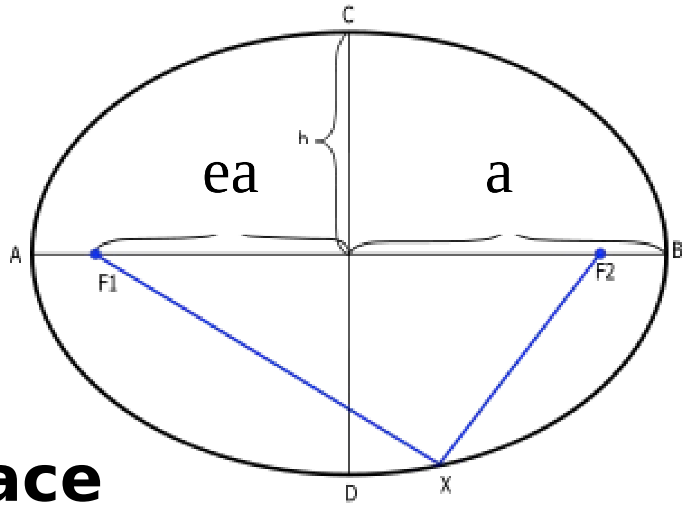
- 3 Dimensional, Second Order D.E.
  - 6 Numbers
  - initial position: r = [x, y, z] at time =  0
  - initial velocity: rdot = [vx, vy, vz] at time = 0
- Each set of initial conditions has unique orbit.  Can characterize orbit with *any* six numbers that will describe it.
- Astronomers use "Orbital Elements" to specify and describe orbits.

# Orbital Elements

- **Size and Shape of Orbit**

  Semimajor Axis - a

  Eccentricity - e



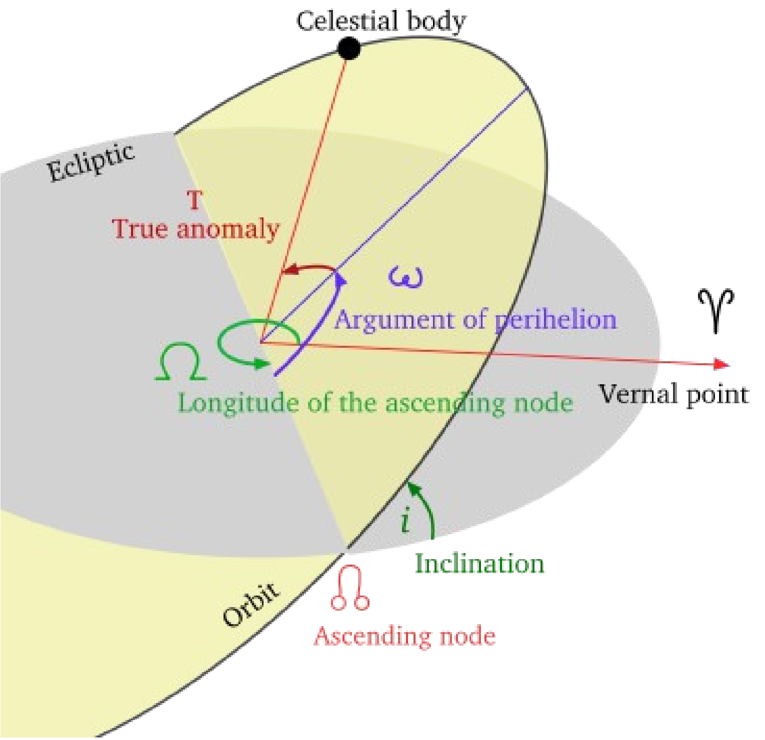

- **Orientation of Orbital Plane in Space**

  Inclination wrt Ecliptic

  Longitude of Ascending Node

  Argument of Perihelion

- **Time of Perihelion Passage**



# Constants of the Motion

Specific Energy (Energy per mass) Conserved

$$\epsilon = -\frac{\mu}{r} + \frac{1}{2} v^2$$

$$\mu = GM_{\odot}$$

Potential Kinetic

Specific Angular Momentum Conserved

$$\vec{h} = \vec{r} \times \vec{v}$$

# Orbital Elements and Constants of the Motion

Semimajor Axis - a

$$a = -\frac{\mu}{2\epsilon}$$

Semimajor Axis depends only on energy

Eccentricity - e

$$e = \left| \frac{\vec{v} \times \vec{h}}{\mu} - \frac{\vec{r}}{|\vec{r}|} \right|$$

Eccentricity depends on angular momentum and energy

Note: Eccentricity Vector above points towards periapsis