## 2) Planet Period vs. Planet Mass

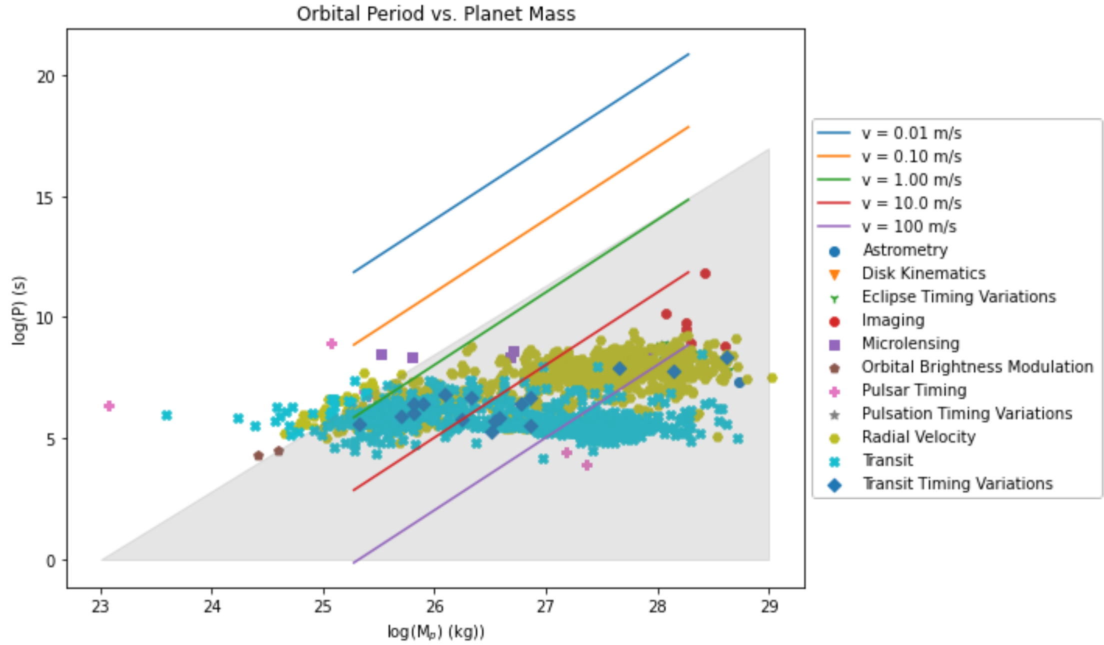
In [652]:
```
# Import relevant modules/packages
import numpy as np
import matplotlib.pyplot as plt
from astropy import units as u
from astropy import constants as const
from matplotlib.patches import Rectangle, Polygon
```

In [322]:
```
# Function to read text files
def Read(filename,skip=97):
    # skip: how many lines before column headers
    # returns: dictionary of data in text file (can access data['column_name'])

    # Read and return data of confirmed exoplanets from NASA Exoplanet Archive
    data = np.genfromtxt(filename,dtype=None,delimiter=',',skip_header=skip,name
s=True,invalid_raise=False,encoding=None)
    return data
```

In [912]:
```
# Define important constants
MJupiter = const.M_jup.value
MSun = const.M_sun.value
G = const.G.value

# Define set of velocities
velocities = [0.01,.1,1,10,100] # in m/s

#Make a list of masses (.1,1,10 Mjup)
mass = [x*MJupiter for x in velocities]
masses = []
```

In [913]:
```
# Define empty list of periods
periods = []

# Function to calculate period from planet mass and orbital velocity
def Period(m,v):
    P = (2*np.pi*G/(v**3))*((m**3)/((MSun+m)**2))
    return P

# Loop to calculate period for each mass
for v in velocities:
    for m in mass:
        masses.append(m)
        periods.append(Period(m,v))

# Create figure and axis object
plt.figure(figsize=(10,6))
ax = plt.subplot(111)

# Plot periods vs. masses of artifical data
ax.plot(np.log10(masses[0:4]),np.log10(periods[0:4]),label='v = 0.01 m/s')
ax.plot(np.log10(masses[5:9]),np.log10(periods[5:9]),label='v = 0.10 m/s')
ax.plot(np.log10(masses[10:14]),np.log10(periods[10:14]),label='v = 1.00 m/s')
ax.plot(np.log10(masses[15:19]),np.log10(periods[15:19]),label='v = 10.0 m/s')
ax.plot(np.log10(masses[20:24]),np.log10(periods[20:24]),label='v = 100 m/s')

# Extract data from NASA exoplanet archive
data = Read('table.csv')

# Define x and y datasets from archive to plot
xdata = []
ydata = []

# Define list of symbols to use for different discovery methods
markers = ['o','v','1','8','s','p','P','*','H','X','D']

# Loop through discovery methods
for i in range(0,len(np.unique(data['discoverymethod']))):

    # Identifty locations of data points for each discovery method
    detections = np.where(data['discoverymethod'] == np.unique(data['discoverymet
hod'])[i])

    # Define x (mass) and y (semi-major axis) datasets to plot
    x = [x*u.M_earth.to(u.kg) for x in data['pl_bmasse'][detections]]
    y = [y*u.d.to(u.s) for y in data['pl_orbper'][detections]]

    # Add x and y list for each method to a master list
    xdata.append(x)
    ydata.append(y)

    # Make a scatter plot for each datset with a new symbol for each discovery me
thod
    ax.scatter(np.log10(xdata[i]),np.log10(ydata[i]),label=np.unique(data['discov
erymethod'])[i],marker=markers[i])

# Flatten x and y lists for shading
xflat = [y for x in xdata for y in x]
yflat = [y for x in ydata for y in x]

# Set axis labels, title, and legend (legend outside of figure)
ax.set_title('Orbital Period vs. Planet Mass')
ax.set_xlabel(r'log(M$_{p}$ (kg))')
ax.set_ylabel('log(P) (s)')
ax.legend(loc='center left',bbox_to_anchor=(1, 0.5))
```

```
/usr/local/Anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:5: Conversi
onWarning: Some errors were detected !
    Line #760 (got 11 columns instead of 91)
    Line #4226 (got 11 columns instead of 91)
  """
```




## Comments about Problem 2

**Modern spectographs can only observe reflex velocities above 1 m/s, so they are currently not sensitive enough to detect any planets with a less signficant impact on their host star than that. With improved sensitivity, less massive planets can be detected with this method since the threshold for detecting their effect on the reflex veloctity of the host star will be lower.**

## 4) Radiative Equilibrium Temperature vs. Semi-Major Axis

$$T_{eq} = \left(\frac{1-A}{\pi}\right)^{1/4}\sqrt{\frac{R_*}{2a}}T_{eff}$$

### G0V: T=5920K, R=1.12$R_\odot$

### M5V: 3030K, R=.199$R_\odot$

### F0V: 7220K, R=1.79$R_\odot$

### A0V: 9700K, R=2.09$R_\odot$

### G2V: 5770K, R=1.01$R_\odot$

In [76]:

```
# Define value of Solar radius
RSun = const.R_sun.to(u.au).value

# Define min and max liquid water temperatures in K
# Source (slide 3):https://www.astro.umd.edu/~miller/teaching/astr380f09/slides1
4.pdf
TempWaterMin = 274.15
TempWaterMax = 303.15
```

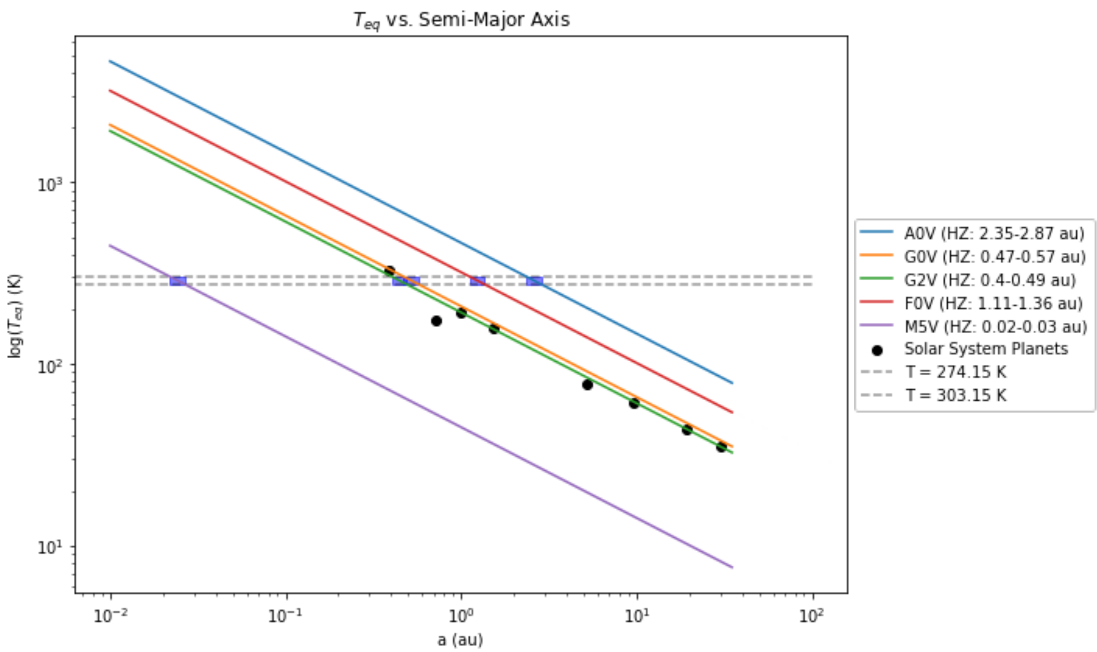
In [87]:

```
# Function to calculate radiative equilibrium temperature for a given host star a
nd semi-major axis
def TeqSMA(startype,SMA):
    # startype: A0V,G0V,G2V,M5V,or F0V
    # albedo: 0-1 (default=0.3)
    # returns: habitable zone range and Teq of planet

    # Define properties of host star
    if startype == 'A0V':
        R = 2.09*RSun # Radius in Rsun
        Teff = 9700 # Effective temperature of host star in K
        albedo = [0.3]*len(SMA)
    elif startype == 'G0V':
        R = 1.12*RSun
        Teff = 5920
        albedo = [0.3]*len(SMA)
    elif startype == 'G2V':
        R = 1.01*RSun
        Teff = 5770
        albedo = [0.3]*len(SMA)
    elif startype == 'M5V':
        R = .199*RSun
        Teff = 3030
        albedo = [0.3]*len(SMA)
    elif startype == 'F0V':
        R = 1.79*RSun
        Teff = 7220
        albedo = [0.3]*len(SMA)
    elif startype == 'Sun':
        R = 1.0*RSun
        Teff = 5780
        # Source wikipedia (references for each value; https://en.wikipedia.org/w
iki/Albedo#Astronomical_albedo)
        albedo = [0.09,0.76,0.31,0.25,0.50,0.34,0.30,0.29]
    else:
        print('Enter valid startype')

    # Calculate minimum and maximum habitable zone distances
    HZMax = R/2*np.sqrt((1-0.3)/np.pi)*((Teff/TempWaterMin)**2)
    HZMin = R/2*np.sqrt((1-0.3)/np.pi)*((Teff/TempWaterMax)**2)

    # Make empty list of Teq
    Teq = []

    # Calculate Teq for each a and add it to list
    for i in range(0,len(SMA)):
        prefactor = ((1-albedo[i])/np.pi)**(1/4)
        Teq.append(prefactor*np.sqrt(R/(2*SMA[i]))*Teff)

    return(HZMax,HZMin,Teq)
```

In [106]:

```
# List names of planets for plotting later
SolSys = ['Mercury','Venus','Earth','Mars','Jupiter','Saturn','Uranus','Neptune']

# Establish semi-major axes of solar system planets (in au)
# Source: https://www.princeton.edu/~willman/planetary_systems/Sol/
aSolSys = [.387,.723,1.00,1.52,5.20,9.54,19.2,30.1]
logaSolSys = [np.log(x) for x in aSolSys]

# Calculate Teq for solar system planets
a,b,TeqSolSys = TeqSMA('Sun',aSolSys)
logTeqSolSys = [np.log10(x) for x in TeqSolSys]
```

In [909]:
```
# Establish figure and axis object for plotting
plt.figure(figsize=(10,6))
ax = plt.subplot(111)

# Make list of semimajor axes|
a = np.linspace(0.01,35,50) # in au

# Make a plot with a line for each startype
startypes=['A0V','G0V','G2V','F0V','M5V']
for i in range(len(startypes)):
    maxD,minD,y = TeqSMA(startypes[i],a)

    # Plot Teq vs. a with habitable zones shaded
    string = ' (HZ: {0}-{1} au)'.format(np.round(minD,2),np.round(maxD,2))
    ax.add_patch(Rectangle((minD,TempWaterMin),maxD-minD,TempWaterMax-TempWaterMi
n,fill=True,color='b',alpha=0.5))
    ax.loglog(a,y,label=startypes[i]+string)

# Plot solar system planets
#for planet in SolSys:
ax.scatter(aSolSys,TeqSolSys,label='Solar System Planets',color='black')

# Illustrate min and max liquid water temps.
plt.hlines(TempWaterMin,-2,100,label='T = {0} K'.format(TempWaterMin),linestyl
e='--',alpha=0.4)
plt.hlines(TempWaterMax,-2,100,label='T = {0} K'.format(TempWaterMax),linestyl
e='--',alpha=0.4)

# Set plot parameters
ax.set_xlabel('a (au)')
ax.set_ylabel(r'log($T_{eq}$) (K)')
ax.set_title(r'$T_{eq}$ vs. Semi-Major Axis')
ax.legend(loc='center left',bbox_to_anchor=(1, 0.5))
plt.tight_layout()
```



**The planets in our solar system fall very close to the G2V line which verifies that the Sun is a G2V-type star.**

## 5) Angular Shift of Stellar Host vs. Planet Mass (Astrometric Motion)

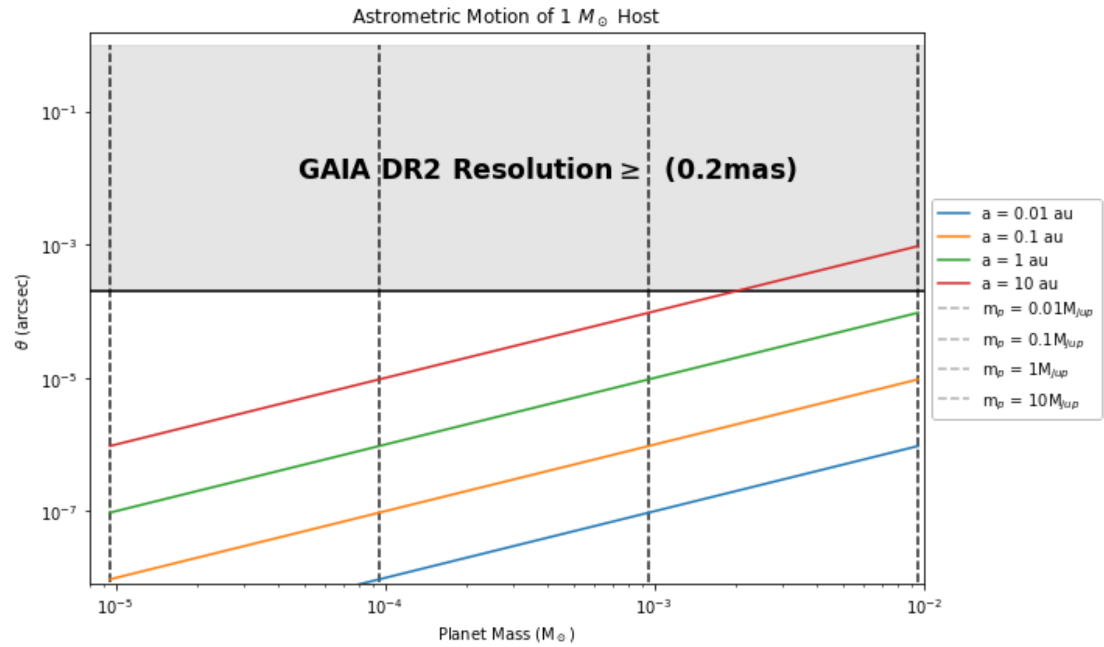
**i) For 2-body orbit w/ origin at CoM:** $\frac{m_1}{m_2} = \frac{a_2}{a_1}$ **-->** $\frac{m_*}{m_p} = \frac{a_p}{a_*}$

**ii) Angular shift (**$\theta$**) corresponds to parallax:** $\theta = \frac{a_*}{D}$ **where D is the distance to the system**

**iii) We are to choose 4 different** $a_p$ **values and loop over different masses to calculate values for** $a_*$ **and thus** $\theta$

In [726]:
```
# Define set of planet semi-major axes
a_planet = [0.01,.1,1,10] # in au

# Define distance to system
D = 100*u.pc.to(u.au)
print(D)

#Make a list of masses (.01,.1,1,10 MJup in terms of MSun)
m_planet = [x*const.M_jup.to(u.Msun).value for x in a_planet]

# Empty list-of-lists of masses for plotting purposes
masses = [[],[],[],[]]

# Make empty list-of-lists of angular shifts
theta = [[],[],[],[]]

# Create figure and axis object
plt.figure(figsize=(10,6))
ax = plt.subplot(111)

# Iterate over all planet mass and SMA combinations to calculate theta for each
for i in range(len(a_planet)):
    for m_p in m_planet:

        # Calculate star SMA from 2-body orbit ratio (m1/m2=a2/a1)
        a_star = a_planet[i]*m_p
        shift = (a_star/D)*206265 # convert from radians to arcsec

        # Add mass and angular shift (in mas) to lists
        masses[i].append(m_p)
        theta[i].append(shift)

    # Plot theta vs. mass
    ax.loglog(masses[i],theta[i],label='a = {0} au'.format(a_planet[i]))

    # Plot lines for important masses
    ax.vlines(masses[i],0,1,label=r'm$_{p}$ = '+'{0}'.format(a_planet[i])+r'M$_{j
up}$',linestyle='--',alpha=0.3)

# Include other plot features
title = r'Astrometric Motion of 1 $M_\odot$ Host'
ax.set_title(title)
ax.set_xlabel(r'Planet Mass (M$_\odot)$')
ax.set_ylabel(r'$\theta$ (arcsec)')
ax.legend(loc='center left',bbox_to_anchor=(1, 0.5))
ax.set_xlim(8*10**-6,10**-2)
ax.set_ylim(8*10**-9,1.5)

# Show GAIA resolution (Source: https://www.cosmos.esa.int/web/gaia/dr2)
ax.add_patch(Rectangle((0,0.2/1000),.01,1,color='gray',alpha=0.2))
GAIA_label = r"  GAIA DR2 Resolution$\geq$  (0.2mas)"
ax.text(4*10**-5,10/1000,GAIA_label,fontsize='xx-large',fontweight='bold')
ax.hlines(0.2/1000,8*10**-6,10**-2)
plt.tight_layout()
```

```
20626480.624548033
```




**Assuming i=0deg for this problem is not necessary because you could still observe the angular shift of the stellar host at any inclination. The star would still move the same amount, it would just appear to take longer or shorter to shift that amountdepending on the system's inclination**

## 6) Minimum Planet Mass (solving mass function)

**i) Mass Function:** $\frac{m_2^3}{(m_1+m_2)^2} sin^3 i = \frac{P}{2\pi G} v_{1r}^3$

$m_2$**: planet mass,** $m_1$**: host star mass,** $i$**: inclination,** $P$**: orbital period,** $v_{1r}$**: host star reflex velocity**

**ii) Known values:** $P$**=3.0 days,** $m_1 = 1M_\odot$**,** $v_{1r} = 50m/s$ **(amplitude is 100 m/s)**

**iii) Minimum mass corresponds to i=90; right side is a constant**

**iv) Loop through** $m_2$**'s to plug into left side to match right**

In [458]:
```
# Define constants on right side of mass function
m1 = (1.0*u.M_sun).to(u.kg) # host star mass in kg
v1 = 50*u.m/u.s # host star reflex velocity in m/s
P = (3.0*u.d).to(u.s) # orbital period of planet in s
G = const.G # gravitational constant in mks

# Calculate right side of mass function
right = P/(2*np.pi*G)*(v1**3)
print(right)
print(right.to(u.Msun))
```

```
7.726083868236984e+19 kg
3.8855589997270856e-11 solMass
```

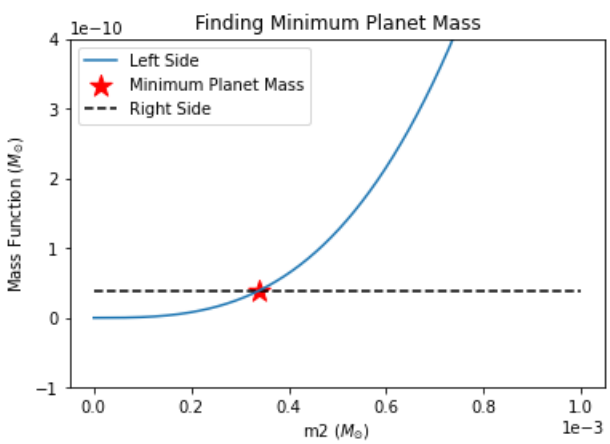
In [300]:

```
# Visualize mass function and value of right side
rightMsun = right.to(u.M_sun).value

# Choose minimum and maximum m2
m2min = 0.0
m2max = 0.001
numSteps = 5000
m2_test = np.linspace(m2min,m2max,numSteps) # in Msun
increment = m2max/numSteps

# Create array to save values of left side for each m2
lefts = []

# Loop over m2's
for m2 in m2_test:
    # Calculate left side for given m2 and add it to list
    left = (m2**3)/((m1.to(u.M_sun).value+m2)**2)
    lefts.append(left)

    # Calculate and print) ratio of left and right side
    ratio = left/right_line

    # Find m2 that makes ratio=1
    if ratio >= .999 and ratio <= 1.0001:
        print('Success! Minimum planet mass is %.2e Msun'%m2)
        plt.scatter(m2,rightMsun,s=200,color='r',marker='*',label='Minimum Planet
Mass')

# Plot value of left side of mass function vs. m2
plt.plot(m2_test,lefts,label='Left Side')

# Plot value of right side
plt.hlines(right_Msun,m2min,m2max,label='Right Side',linestyle='--')

# Set plot features
plt.ylim(-1e-10,4e-10)
plt.ticklabel_format(axis='x',style='sci',scilimits=(0,0))
plt.xlabel(r'm2 ($M_{\odot}$)')
plt.ylabel(r'Mass Function ($M_{\odot}$)')
plt.title('Finding Minimum Planet Mass')
plt.legend()
```

```
Success! Minimum planet mass is 3.39e-04 Msun
```

Out[300]:

```
<matplotlib.legend.Legend at 0x7f00bc2caa90>
```



### Solving Mass Function Analytically

([https://math.vanderbilt.edu/schectex/courses/cubic/ (https://math.vanderbilt.edu/schectex/courses/cubic/)](https://math.vanderbilt.edu/schectex/courses/cubic/))

### i) General cubic form: $ax^3 + bx^2 + cx + d = 0$

### ii) Solutions are:

$$x = (q + (q^2 + (r - p^2)^3)^{1/2})^{1/3} + (q - (q^2 + (r - p^2)^3)^{1/2})^{1/3} + p$$

### where $p = \frac{-b}{3a}$; $q = p^3 + \frac{bc-3ad}{6a^2}$; $r = \frac{c}{3a}$

### iii) Matching form with mass function: $m_2^3 - \alpha m_2^2 - 2\alpha m_1 m_2 - \alpha m_1^2 = 0$

### in this case: $\alpha = \frac{P}{2\pi G} v_{1r}^3 \Big| x = m_2, a = 1, b = -\alpha, c = -2\alpha m_1,$ $d = -\alpha m_1^2$

In [472]:

```
# Define constants from cubic equation
a = 1.0
b = -rightMsun
c = -2.0*rightMsun
d = -rightMsun

# Function to solve cubic equation
def CubicSolver(a,b,c,d):
    # Define substitutions for solving cubic equation
    p = -b/(3.0*a)
    q = (p**3.0)+((b*c)-(3.0*a*d))/(6.0*(a**2))
    r = c/(3.0*a)

    # Calculate solution
    firstTerm = (q+np.sqrt((q**2)+((r-(p**2))**3)))**(1/3)
    secondTerm = (q-np.sqrt((q**2)+((r-(p**2))**3)))**(1/3)
    analytic_solution = firstTerm+secondTerm+p
    print('The analytical solution is also %.2e Msun!'%analytic_solution)
CubicSolver(a,b,c,d)
```

```
The analytical solution is also 3.39e-04 Msun!
```

# 7) Plotting Binary Orbits

### i) Need to solve Kepler equation numerically: $E - esin(E) = \frac{2\pi}{P}(t - t_0)$

### ii) Use Kepler III to find period:

$$P^2 = \frac{4\pi^2 a^3}{G(m_1+m_2)} = \frac{4\pi^2 a^3}{M}$$

### iii) Say $t_0 = 0$ and use Newton-Raphson method to converge on E:

$$f(E) = E - esin(E) - \frac{2\pi}{P}t$$

$$f'(E) = 1 - ecos(E)$$

In [896]:
```
# Function to numerically solve differentiable equation
# Resource that helped me: https://www.math.ubc.ca/~pwalls/math-python/roots-opti
mization/newton/
def NewtonRaphson(f,df,x0,precision,numSteps):
    # Inputs:
    #    f: function to evaluate
    #    df: derivative of function
    #    x0: initial guess at solution
    #    precision: answer won't exactly be 0, so set a tolerance
    #    numSteps: maximum number of times to iterate

    # Establish first guess at solution
    x = x0

    # Iterate over number of steps
    for i in range(0,numSteps):

        # Evaluate function
        func = f(x)

        # If f(x) is within precision, declare that value of x as the solution
        if abs(func) <= precision:
            #print('A solution of {0:.2e} was found in {1} iterations'.format(x,
i))
            return x

        # If f(x) is not within precision, continue searching for solution
        elif abs(func) > precision:

            # Evaluate derivative
            deriv = df(x)

            # Adjust guess of solution by subtracting quotient of function and de
rivative from the last x
            x -= func/deriv
```

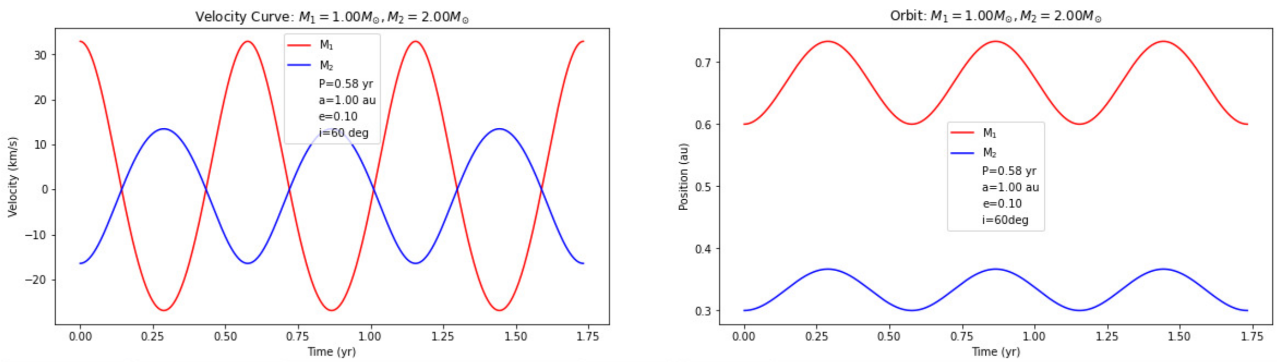
In [895]:
```
# Function and derivative to test NewtonRaphson function
f = lambda x: x**2 - 2
df = lambda x: 2*x
NewtonRaphson(f,df,.1,1e-6,100)
```

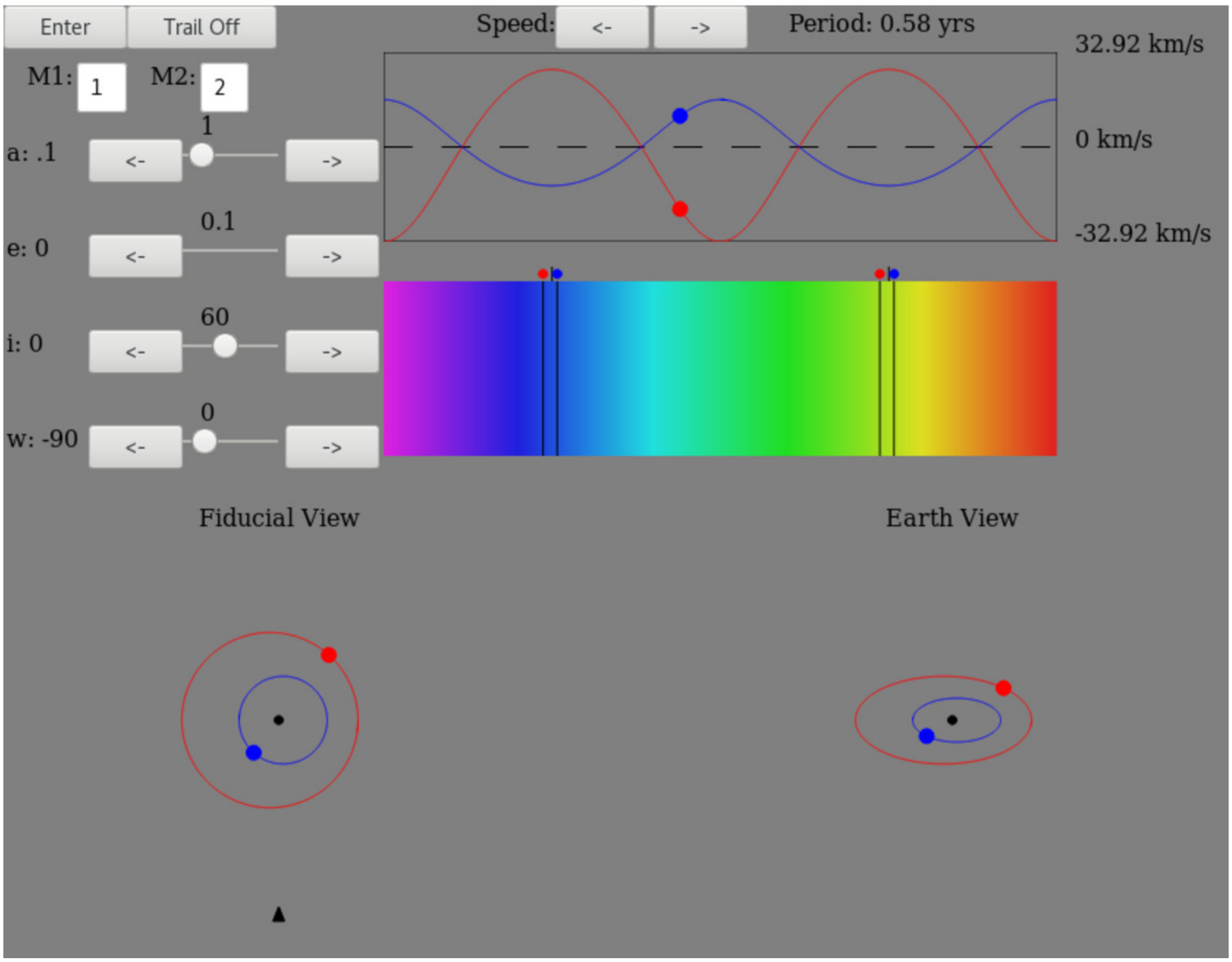
```
A solution of 1.41e+00 was found in 7 iterations
```

Out[895]:
```
1.4142136001158032
```

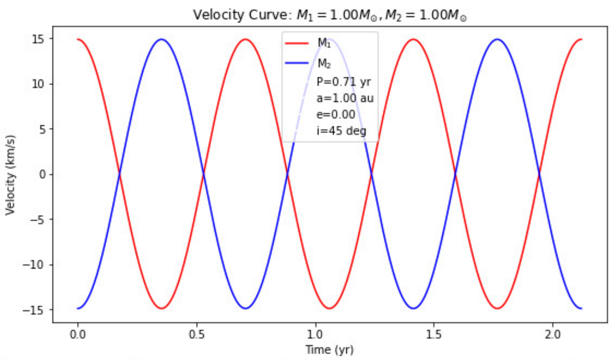
In [884]:
```
# Function to calculate properties of binary orbit
def Orbit(m1,m2,a,eccen,i):
    # Inputs:
    #     m1 = mass of 1st body in system (in Msun)
    #     m2 = mass of 1nd body in system (in Msun)
    #     a = system semi-major axis (in au; a=a1+a2)
    #     e = eccentricity of orbits (0<e<1)
    #     i = inclination of binary system (0 to 90 deg)
    #     precision = precision of solution found by Newton-Raphson method
    # Returns:
    #     velocity curve and astrometric orbit data (times,velocities,positions)

    # Convert G, m, a, and i to appropriate units; then calculate M
    G = const.G.to(u.au**3/(u.Msun*u.yr**2))
    mSolar = (m1+m2)*u.M_sun
    a = a*u.au
    i *= np.pi/180 # convert inclination to radians for numpy
    M = G*mSolar # in au^3/yr^2

    # Find period (in yr)
    P = np.sqrt(4*(np.pi**2)*(a**3)/M).value

    # Find individual semi-major axes (use m1/m2=a2/a1)
    a1 = a.value/(1+m1/m2)
    a2 = a.value-a1

    # Define set of times to evaluate E over (~0 to 3*Period)
    times = np.linspace(0.001,3*P,1000)

    # Define empty list of eccentric and true anomalies to fill
    EA_list = []
    TA_list = []

    # Define empty list of positions and velocities for each body
    r1_list = []
    r2_list = []
    v1_list = []
    v2_list = []

    # Calculate first term of E and theta relationship
    term1 = ((1+eccen**2)/(1-eccen**2))**(1/2)

    # Find E at each time
    for time in times:

        # Define Kepler equation and its derivative
        f = lambda E: E - eccen*np.sin(E) - (2*np.pi/P)*time
        df = lambda E: 1 - eccen*np.cos(E)

        # Use NewtonRaphson function to find E at each time
        EA = NewtonRaphson(f,df,1e-2,1e-6,100)
        EA_list.append(EA)

        # Convert eccentric anomaly (eA) to true anomaly (theta)
        TA = 2*np.arctan(term1*np.tan(EA/2))
        TA_list.append(TA)

        # Calculate velocities at each time
        v1 = (2*np.pi*a1*np.sin(i))/(P*np.sqrt(1-eccen**2))*(np.cos(TA)+ecce
n)*(u.au/u.yr).to(u.km/u.s)
        v2 = (2*np.pi*a2*np.sin(i))/(P*np.sqrt(1-eccen**2))*(np.cos(TA)+ecce
n)*(u.au/u.yr).to(u.km/u.s)
        v1_list.append(v1)
        v2_list.append(-v2)
```

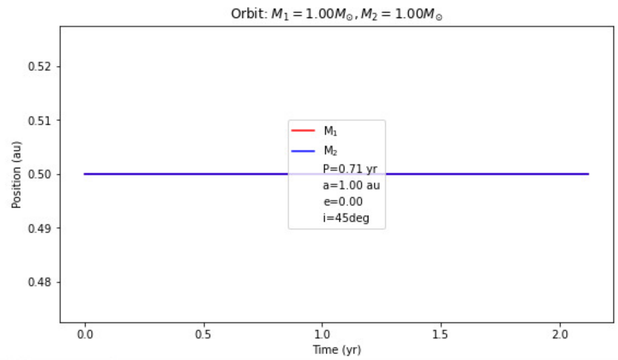
In [ ]:
```
Orbit(1,2,1,0.1,60)
```

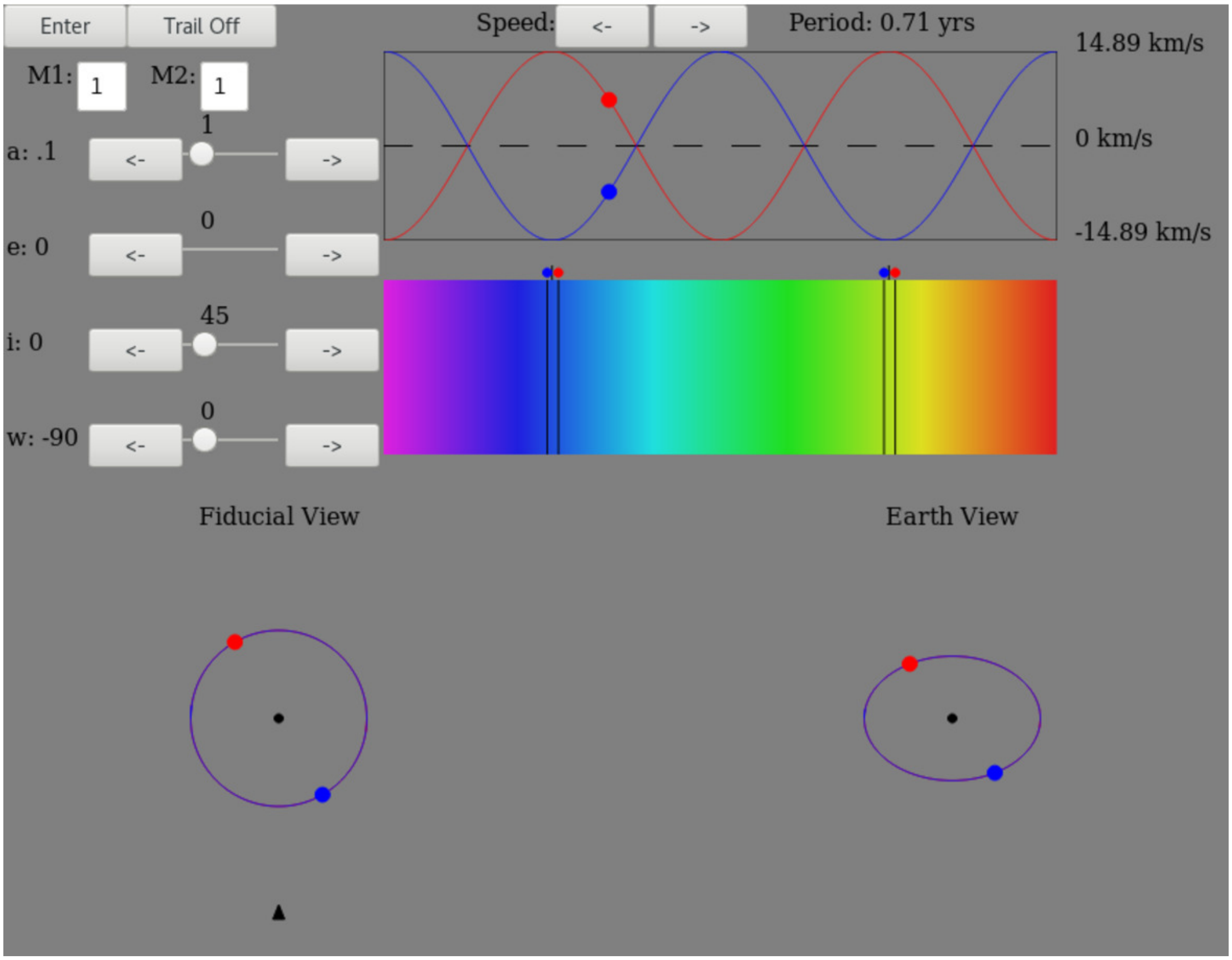


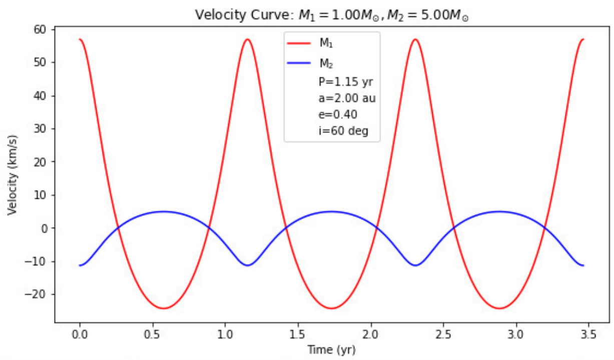

In [ ]:
```
Orbit(1,1,1,0.0,45)
```

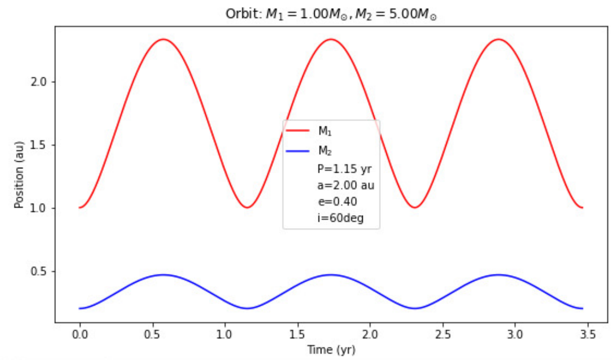


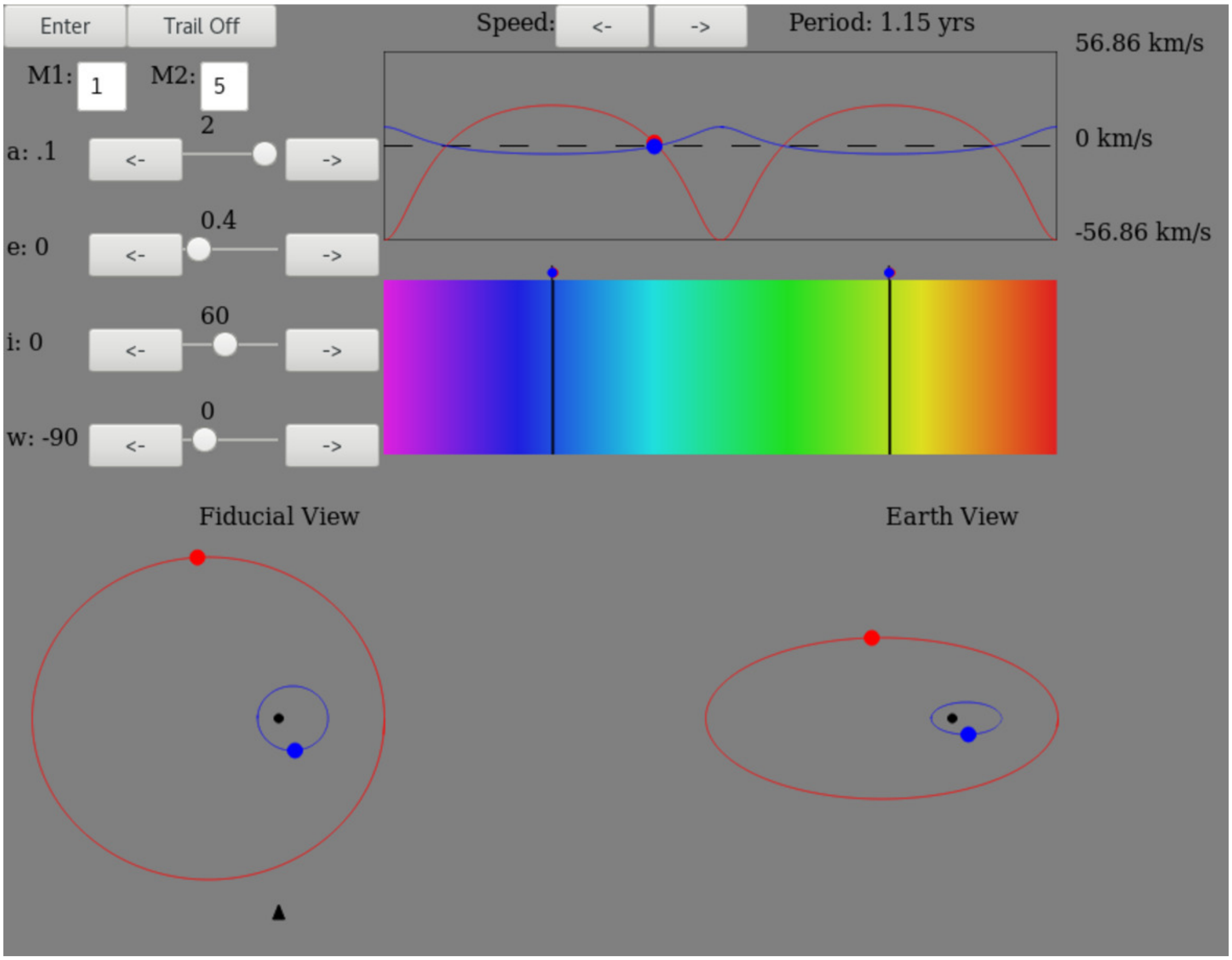




In [ ]:
```
Orbit(1,5,2,0.4,60)
```


Velocity Curve: $M_1 = 1.00M_{\odot}, M_2 = 5.00M_{\odot}$
Velocity (km/s)
Time (yr)
$M_1$
$M_2$
P=1.15 yr
a=2.00 au
e=0.40
i=60 deg


Orbit: $M_1 = 1.00M_{\odot}, M_2 = 5.00M_{\odot}$
Position (au)
Time (yr)
$M_1$
$M_2$
P=1.15 yr
a=2.00 au
e=0.40
i=60deg


Enter
Trail Off
Speed:
<-
->
Period: 1.15 yrs
56.86 km/s
0 km/s
-56.86 km/s
M1: 1
M2: 5
a: .1
2
e: 0
0.4
i: 0
60
w: -90
0
Fiducial View
Earth View

In [ ]: