In [36]:

```python
import numpy as np
import matplotlib.pyplot as plt
```

Lecture 3 - 2 and 3 dimensional kinematics

In this lecture we look at vectors in 2 and 3 dimensional motion and study a specific example of 2 dimensional motion - projectile motion.

Vectors and Scalars

Vector quantities with number, **direction** and units:

- Displacement $\vec{r}$ [m]
- Velocity $\vec{v}$ [ms^{-1}]
- Acceleration $\vec{a}$ [ms^{-2}]

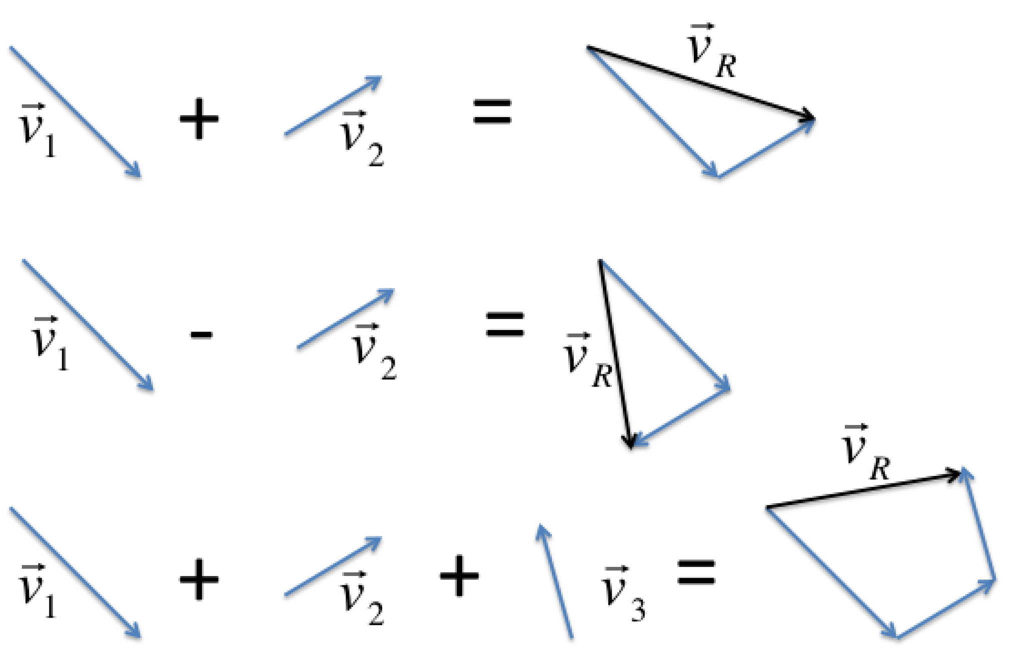
Scalar quantities number and units only

- Distance traveled [m]
- Speed [ms^{-1}]

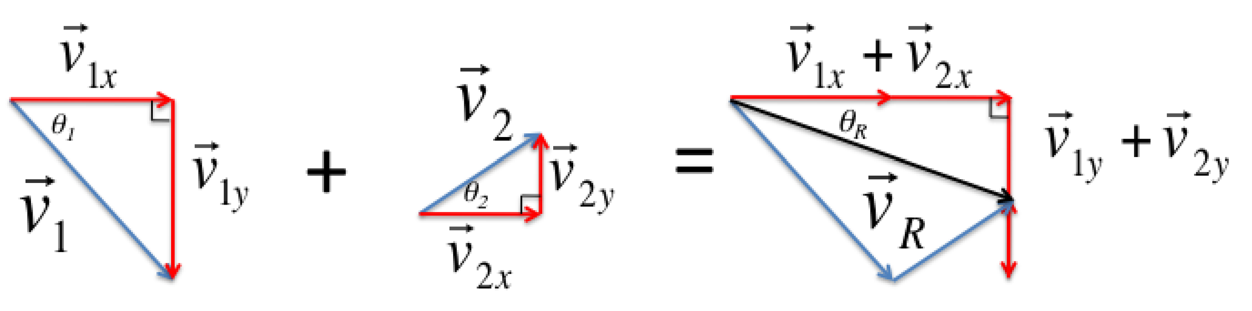
Graphical representation of vectors and components

It is frequently useful to draw two dimensional vectors as arrows, and to split them in to components that lie along the coordinate axes. The choice of coordinate axes is up to you. But choosing the right ones will make the problem easier or harder.

Adding and subtracting vectors



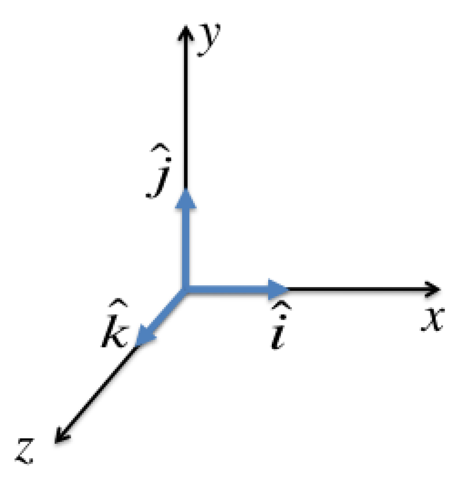
Vectors Components



$v_{1x} = v_1 \cos\theta_1$	$v_{2x} = v_2 \cos\theta_2$
$v_{1y} = v_1 \sin\theta_1$	$v_{2y} = v_2 \sin\theta_2$

$v_{Rx} = v_{1x} + v_{2x}$	$\tan\theta_R = \frac{v_{Ry}}{v_{Rx}}$
$v_{Ry} = v_{1y} + v_{2y}$	$v_R = \sqrt{v_{Rx}^2 + v_{Ry}^2}$

Unit Vectors



It can be useful to express vector quantities in terms of unit vectors. These are dimensionless vectors of length = 1 that point along the coordinate axes. They are usually denoted with carets (hats), i.e. $(\hat{\imath}, \hat{\jmath}, \hat{k})$

For example:

$\vec{v}, \mathrm{ms}^{-1} = v_x, \mathrm{ms}^{-1}, \hat{\imath} + v_y, \mathrm{ms}^{-1}, \hat{\jmath} + v_z, \mathrm{ms}^{-1}, \hat{k}$

or

$\vec{r}, \mathrm{m} = x, \mathrm{m}, \hat{\imath} + y, \mathrm{m}, \hat{\jmath} + z, \mathrm{m}, \hat{k}$

Vectors and motion

$\vec{r_1} = x_1\,\hat{\imath} + y_1\,\hat{\jmath} + z_1\,\hat{k}$

$\vec{r_2} = x_2\,\hat{\imath} + y_2\,\hat{\jmath} + z_2\,\hat{k}$

$\Delta\vec{r} = \vec{r_2} - \vec{r_1} = (x_2 - x_1)\,\hat{\imath} + (y_2 - y_1)\,\hat{\jmath} + (z_2 - z_1)\,\hat{k}$

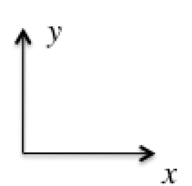
Average velocity: $\vec{v}_{ave} = \frac{\Delta\vec{r}}{\Delta t}$

Instantaneous velocity: $\vec{v} = \frac{d\vec{r}}{dt} = \frac{dx}{dt}\,\hat{\imath} + \frac{dy}{dt}\,\hat{\jmath} + \frac{dz}{dt}\,\hat{k} = v_x\,\hat{\imath} + v_y\,\hat{\jmath} + v_z\,\hat{k}$

Average acceleration: $\vec{a}_{ave} = \frac{\Delta\vec{v}}{\Delta t}$

Instantaneous acceleration: $\vec{a} = \frac{d\vec{v}}{dt} = \frac{dv_x}{dt}\,\hat{\imath} + \frac{dv_y}{dt}\,\hat{\jmath} + \frac{dv_z}{dt}\,\hat{k} = \frac{d^2x}{dt^2}\,\hat{\imath} + \frac{d^2y}{dt^2}\,\hat{\jmath} + \frac{d^2z}{dt^2}\,\hat{k}$

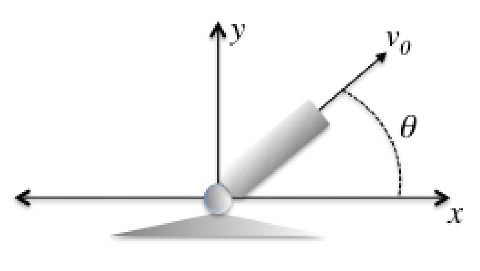
Projectile motion



If g = $9.8ms^{-2}$

$a_x = 0$	$a_y = -g$
$v_x = v_{x0}$	$v_y = v_{y0} - gt$
$x = x_0 + v_{x0}t$	$y = y_0 + v_{y0}t - \frac{1}{2}gt^2$

Projectile motion for a "cannon"



$a_x = 0$	$a_y = -g$
$v_x = v_0\cos\theta$	$v_y = v_0\sin\theta - gt$
$x = x_0 + v_0\cos\theta t$	$y = y_0 + v_0\sin\theta t - \frac{1}{2}gt^2$

Motion path

Taking x_0,y_0 as $(0,0)$

$t = \frac{x}{v_0\cos\theta}$

$y = \frac{\sin\theta}{\cos\theta}x - \frac{g}{(2v_0^2\cos^2\theta)}x^2$

Horizontal Range → Solve for $y = 0$

$0 = x(\frac{\sin\theta}{\cos\theta} - \frac{g}{(2v_0^2\cos^2\theta)}x)$, which has two solutions

$x = 0$ and $x = \frac{2v_0^2\sin\theta\cos\theta}{g} \rightarrow R = \frac{2v_0^2\sin\theta\cos\theta}{g}$

Angle for which maximum range is achieved

$R = \frac{2v_0^2\sin\theta\cos\theta}{g} = \frac{v_0^2}{g}\sin 2\theta$

$\frac{dR}{d\theta} = \frac{2v_0^2}{g}(\frac{d(\sin\theta)}{d\theta}\cos\theta) + \sin\theta\frac{d(\cos\theta)}{d\theta})$

$= \frac{2v_0^2}{g}(\cos^2\theta - \sin^2\theta) = \frac{2v_0^2}{g}\cos 2\theta$

or

$\frac{dR}{d\theta} = \frac{v_0^2}{g}\frac{d(\sin 2\theta)}{d\theta}\frac{d(2\theta)}{d\theta} = \frac{2v_0^2}{g}\cos 2\theta$

$\frac{dR}{d\theta} = 0$ at $\theta = 45^o, 135^o$

45^o and 135^o correspond to maximum range.

Predict range for straight shot

$$y = \frac{\sin\theta}{\cos\theta}x - \frac{g}{(2v_0^2\cos^2\theta)}x^2$$

In [38]:

```python
# straight shot angle is zero
y = 1.2 #m
v0 = 5 #m/s
g =9.8 #m/s2
x = np.sqrt(y*(2*v0**2)/g)
print(x)
```

```
2.4743582965269675
```

Relative velocity

As in 1 dimensional motion, to find the velocity of an object relative to a moving reference frame, subtract the reference frame velocity from that of the object. Of course you have to do this correctly observing vector rules! Usually you are going to have to break things down in to appropriate components. Several of the homework problems are relative velocity problems and we'll talk more about this in the next lecture.

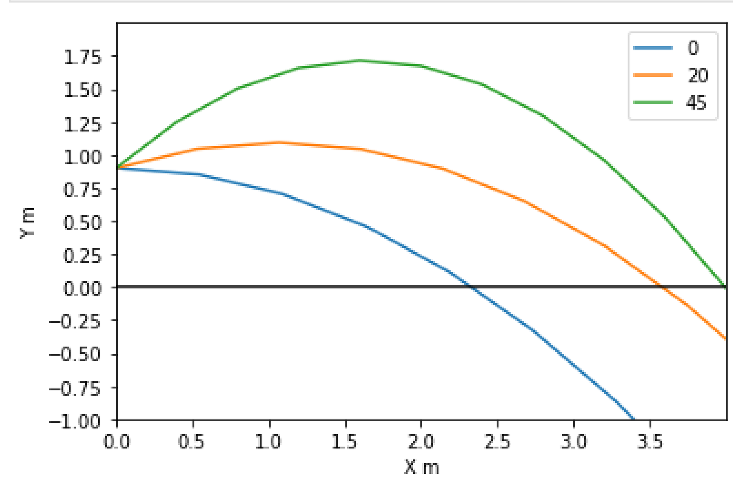
In class cannon experiment

We made 3 shots, at 3 angles (0,20 and 45)^o and measured the initial velocity (5.466,5.7,5.656)m/2. We also measured the range of the first shot which was ~2.2 m. The cannon was placed at x0=0, y0=90 cm (we measured this approximately)

The next code cell shows the calculation of the exact trajectories using the values of the initial velocity.

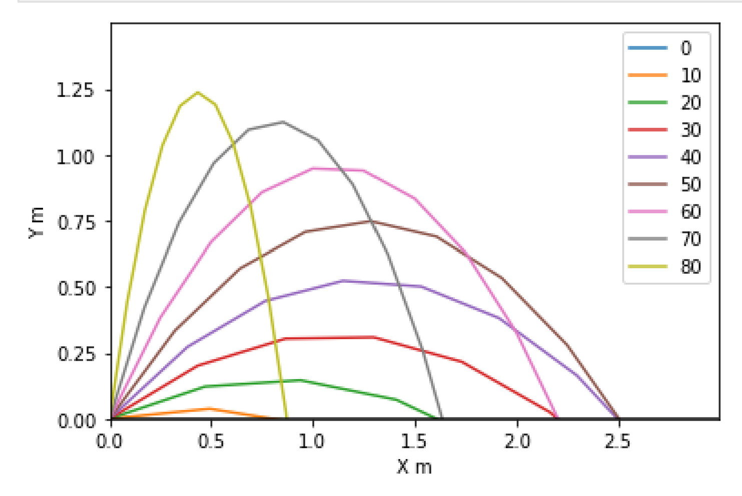
In [69]:

```python
# Experiment done in class: 3 shots
# Define variables
v0 = [5.466,5.7,5.656] # array with 3 velocities
angles = [0,20,45] # array with three shot angles
t = np.arange(0,1.5,0.1) # time discretization, 0-1.5 s in 0.1 s steps
x0 = 0 #m
y0 = 0.9 #m
deg2rad = np.pi/180# convert to radians
for i in range(3):
   theta = angles[i]
   name = str(theta) # label for the plots
   theta = theta * deg2rad
   g = 9.8 #m/s2
   x = x0 + v0[i]*np.cos(theta)*t
   y = y0 + v0[i]*np.sin(theta)*t - 0.5*g*t**2
   plt.plot(x,y, label=name)
   plt.legend()

# Plot a horizonal line at zero
plt.axhline(y=0, color='k', linestyle='-')
# Set Labels, ranges and Ticks
plt.xlabel("X m")
plt.ylabel("Y m")
plt.xlim([0, 4])
plt.ylim([-1, 2])
plt.xticks(np.arange(0, 4, 0.5))
plt.yticks(np.arange(-1, 2, 0.25))
plt.show()
```



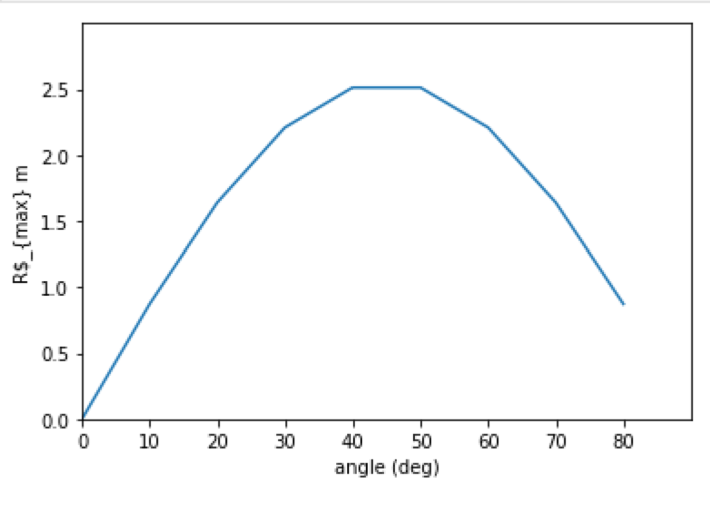
In [73]:

```python
# Same code, now of a larger range of angles
# We also plot the maximum range as a function of the angle
# Using the derived formula above.
# Rmax = (2*v0**2*sin(theta)*cos(theta)) / g
# Define variables
Rmax = []
v0 = 5.0   # m/s array with 3 velocities
angles = [0,10,20,30,40,50,60,70,80] # array with three shot angles
t = np.arange(0,1.5,0.1) # time discretization, 0-1.5 s in 0.1 s steps
x0 = 0 #m
y0 = 0 #m
deg2rad = np.pi/180# convert to radians
for i in range(9):
   theta = angles[i]
   name = str(theta) # label for the plots
   theta = theta * deg2rad
   g = 9.8 #m/s2
   x = x0 + v0*np.cos(theta)*t
   y = y0 + v0*np.sin(theta)*t - 0.5*g*t**2
   #max range
   Rmax.append((2*v0**2*np.sin(theta)*np.cos(theta)) / g)
   plt.plot(x,y, label=name)
   plt.legend()

# Plot a horizonal line at zero
plt.axhline(y=0, color='k', linestyle='-')
# Set Labels, ranges and Ticks
plt.xlabel("X m")
plt.ylabel("Y m")
plt.xlim([0, 3])
plt.ylim([0, 1.5])
plt.xticks(np.arange(0, 3, 0.5))
plt.yticks(np.arange(0, 1.5, 0.25))
plt.show()
```



In [75]:

```python
# Plot R max versus angle
plt.plot(angles, Rmax)
# Set Labels, ranges and Ticks
plt.xlabel("angle (deg)")
plt.ylabel("R$_{max} m")
plt.xlim([0, 90])
plt.ylim([0, 3])
plt.yticks(np.arange(0, 3, 0.5))
plt.xticks(np.arange(0, 90, 10))
plt.show()
```



In [ ]: