Motion in Two Dimensions and Gravity

#4

Serdar ARITAN

Biomechanics Research Group,
Faculty of Sports Sciences, and
Department of Computer Graphics
Hacettepe University, Ankara, Turkey

Numerical Representation of Vectors

In Python a vector is represented by its component form. The vector **a**:

$$\vec{a} = 1\boldsymbol{i} + 2\boldsymbol{j} + 0\boldsymbol{k} = (1\,, 2\,, 0)$$

$$\vec{b} = 2\boldsymbol{i} - 4\boldsymbol{j} + 0\boldsymbol{k} = (2\,, -4\,, 0)$$

is generated by the following command:

```
>>> import numpy as np
>>> a = np.array([1,2,0])
>>> b = np.array([2,-4,0])
```

Addition and Subtraction

```
>>> c = a + b
>>> print(c)
[3 -2 0]
```

You can decide if you want to use a vector in two - or three dimensions. For example, you could instead have defined the vector a as:

```
>>> a = np.array([1,2])
```

But notice that you can **not** add **two vectors** that do not have the **same number of components.**

Scalar Multiplication

Scalar multiplication is similarly naturally implemented:

```
>>> d = 3*a
>>> print(d)
[3 6]
```

Componentwise Operations

Notice that there is room for error because of the way commands are interpreted. For example, if you add a scalar to a vector, this is interpreted as a **componentwise** addition: The scalar is added to **each of the components**:

```
>>> a = np.array([1,2,0])
>>> e = a + 3
>>> print(e)
[4 5 3]
```

Dot Product

The dot product is found by applying the function **dot**, which returns a scalar:

```
>>> f = np.dot(a, b)
>>> print(f)
-6
```

A common application of the dot product is to find the component of a vector $\vec{a}$ along the direction given by a vector $\vec{b}$. In general, $\vec{b}$, is not a unit vector. We therefore first need to find a unit vector in the direction of $\vec{b}$:

$$u_b = \frac{b}{|b|}$$

Dot Product

and the component of a in this direction is given by the dot product:

$$a_b = a.u_b = a.\frac{b}{|b|}$$

Numerically, this is done in exactly the same way:

```
>>> ab = np.dot(a,b)/np.sqrt(np.dot(b,b))
>>> print(ab)
-1.3416407865
```

Notice that we use the relation:

$$|b| = \sqrt{b.b}$$

for the magnitude of b.

Time Sequences of Vectors

We will often work with a sequence of vectors, corresponding to the time evolution of a vector. For example, we may be interested in the position, $\vec{r}$, or the force, ***F***, as a function of time, *t*:

$$\vec{r}(t) \ \textbf{and} \ \vec{F}(t)$$

Numerically, we will have a corresponding sequence of positions or forces at discrete times, t_i :

$$\vec{r}_i = \vec{r}(t_i) \ \textbf{and} \ \vec{F}_i = \vec{F}(t_i)$$

PHYSICS in COMPUTER ANIMATIONS and GAMES

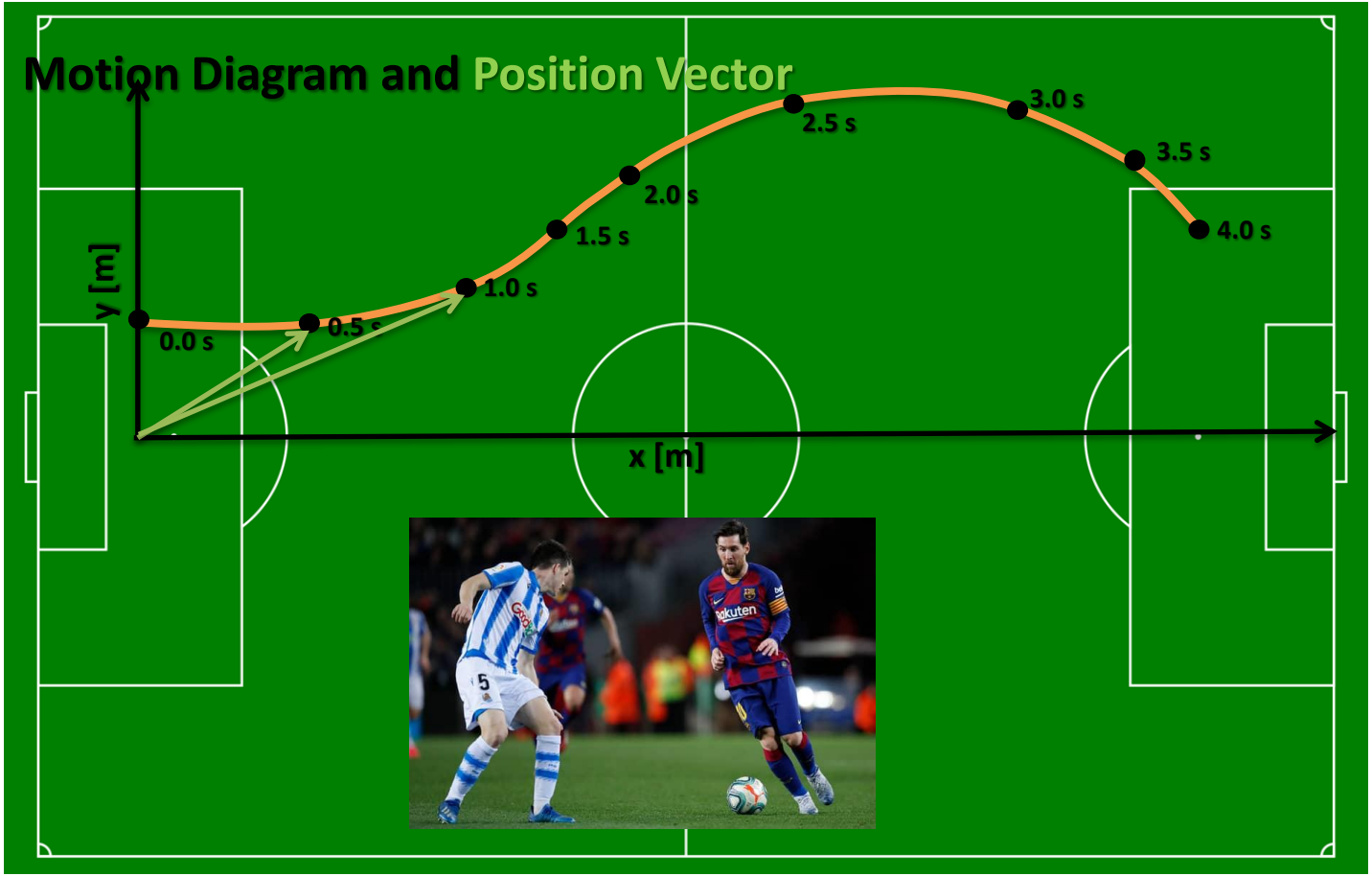
We generate a sequence of *n* vectors r_i with *x*, *y*, and *z* coordinates by:

```
n = 10
r = np.zeros((n, 3),float)
```

We can use mathematical vector operations directly on element in the sequence, as shown in the following example:

```
>>> v = np.array([1.0,-2.0,2.0]) # constant speed
>>> n = 10                       # number of step
>>> r = np.zeros((n,3),float)
>>> r[0] = np.array([0,0,0])     # initial position
>>> dt = 0.1                     # time step
>>> r[1] = r[0] + v*dt           # new position
```

Motion Diagram and Position Vector



We mark the position of the soccer ball regular time intervals and record the positions $\vec{r}(t)$ of the ball relative to the origin at time t_i . We are free to choose the origin and the axes of the coordinate system. The origin determines where we measure the positions from. The position can be decomposed along the x, y, and z-axes respectively

$$\vec{r}(t) = x(t)\boldsymbol{i} + y(t)\boldsymbol{j} + z(t)\boldsymbol{k}$$

where *x(t)*, *y(t)*, and *z(t)* are lengths along the axes and hence have units of length.

For example, the position at $t = 0.5$ s is:

Positions of the soccer ball

t_i (s)	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5
x_i (m)	0.0	16.2	29.9	38.9	46.6	57.2	71.7	84.8
y_i (m)	15.0	14.95	19.0	26.1	31.3	35.4	35.7	30.0

$$\vec{r}(t) = x(t)\boldsymbol{i} + y(t)\boldsymbol{j} + z(t)\boldsymbol{k}$$

$$\vec{r}(0.5) = x(0.5)\boldsymbol{i} + y(0.5)\boldsymbol{j} + z(0.5)\boldsymbol{k} = 16.2\,\boldsymbol{i} + 14.95\,\boldsymbol{j} + 0\,\boldsymbol{k}$$

$$\vec{r}(0.5) = 16.2\,\boldsymbol{i} + 14.95\,\boldsymbol{j}$$

Velocity Vector

The change in position is also a **vector** and is called the **displacement**. The displacement from t = 1.0 s to t = 2.0 s is denoted $\Delta \boldsymbol{r}$(1.0 s):

$$\Delta\vec{\boldsymbol{r}}(1.0) = \vec{\boldsymbol{r}}(2.0) - \vec{\boldsymbol{r}}(1.0) = (46.6\,\boldsymbol{i} + 31.3\,\boldsymbol{j}) - (29.9\,\boldsymbol{i} + 19.0\,\boldsymbol{j})$$

$$\Delta\vec{\boldsymbol{r}}(1.0) = 16.7\,\boldsymbol{i} + 12.3\,\boldsymbol{j}$$

The displacement depends on a difference between two positions, it does not depend on the choice of origin. The **rate of change of the displacement**, the velocity, must therefore also be a **vector**:

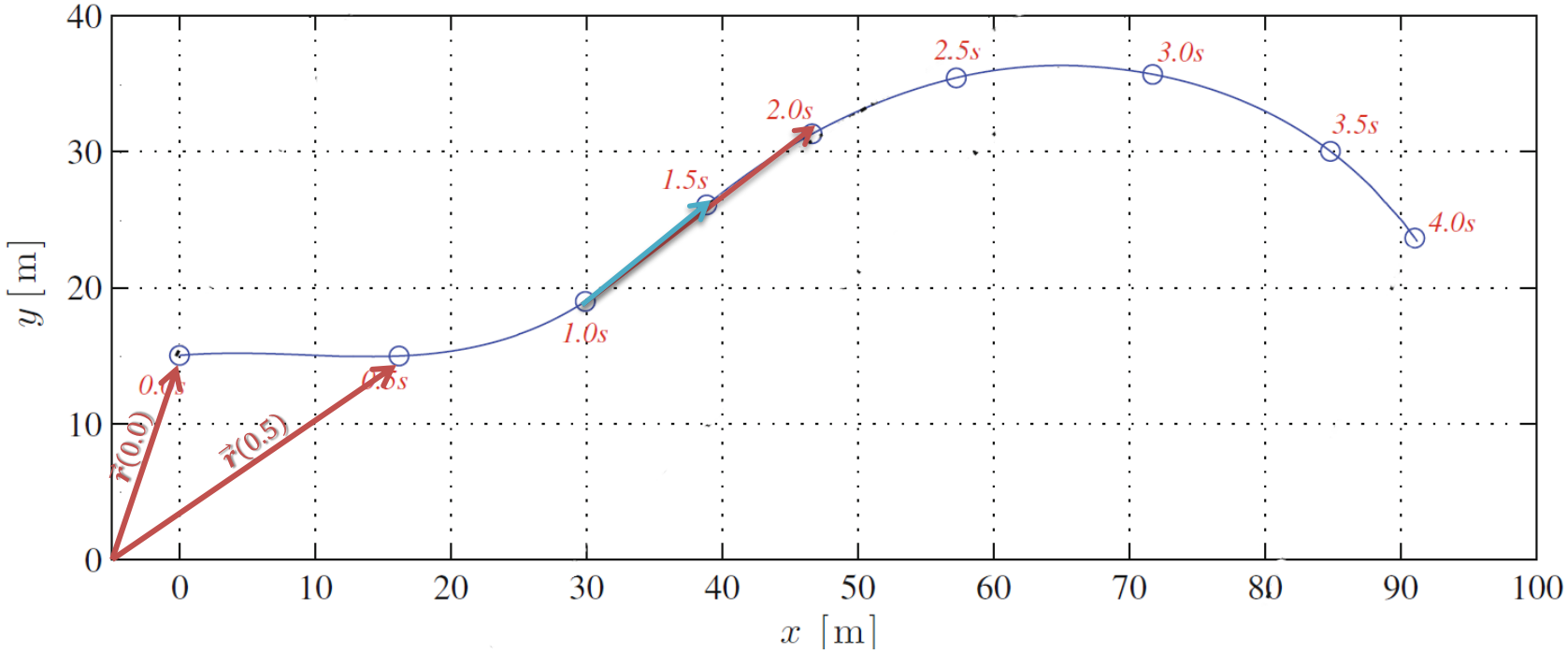
Velocity Vector

The average velocity from a time $t = t_0$ to a time $t = t_0 + \Delta t$ is defined as:

$$\vec{v}(t_0) = \frac{\vec{r}(t_0 + \Delta t) - \vec{r}(t_0)}{\Delta t} = \frac{\Delta \vec{r}}{\Delta t}$$

$$\vec{v}(t_0) = \frac{\Delta \vec{r}}{\Delta t} = \frac{\Delta \vec{r}(1.0\ s)}{1.0\ s} = \frac{16.7\ \boldsymbol{i} + 12.3\ \boldsymbol{j}}{1.0} = 16.7 \frac{m}{s} \boldsymbol{i} + 12.3 \frac{m}{s} \boldsymbol{j}$$

t_i (s)	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5
x_i (m)	0.0	16.2	29.9	38.9	46.6	57.2	71.7	84.8
y_i (m)	15.0	14.95	19.0	26.1	31.3	35.4	35.7	30.0


40
30
20
10
0
y [m]
0
10
20
30
40
50
60
70
80
90
100
x [m]
0.5s
0.0s
1.0s
1.5s
2.0s
2.5s
3.0s
3.5s
4.0s
r(0.0)
r(0.5)

Velocity Vector

If we instead use a time interval Δt = 0.5 s to find the average velocity at t = 1.0 s we find:

$$\Delta\vec{r}(1.0) = \vec{r}(1.5) - \vec{r}(1.0) = (38.9\,\boldsymbol{i} + 26.1\,\boldsymbol{j}) - (29.9\,\boldsymbol{i} + 19.0\,\boldsymbol{j})$$

$$\Delta\vec{r}(1.0) = 9.0\,\boldsymbol{i} + 7.1\,\boldsymbol{j}$$

$$\vec{v}(t_0) = \frac{\Delta\vec{r}}{\Delta t} = \frac{\Delta\vec{r}(1.0\ s)}{0.5\ s} = \frac{9.0\,\boldsymbol{i} + 7.1\,\boldsymbol{j}}{0.5} = 18.0\frac{m}{s}\boldsymbol{i} + 14.2\frac{m}{s}\boldsymbol{j}$$

t_i (s)	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5
x_i (m)	0.0	16.2	29.9	38.9	46.6	57.2	71.7	84.8
y_i (m)	15.0	14.95	19.0	26.1	31.3	35.4	35.7	30.0

The **instantaneous velocity** at the time *t* is defined as the limit of the average velocity when the time interval Δt goes to zero, that is, the time derivative of the position vector, ***r***(t).

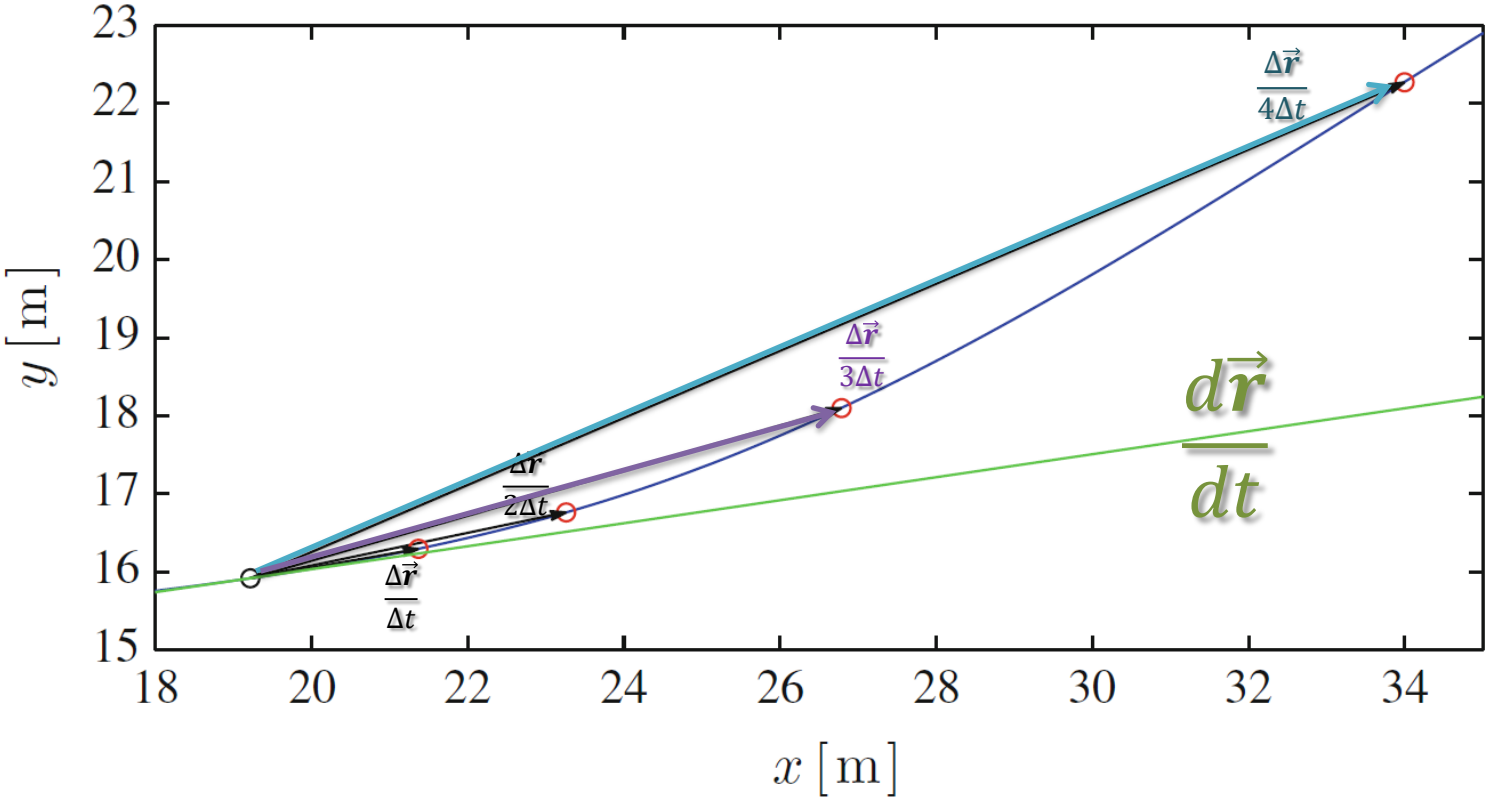
$$\vec{v}(t) = \lim_{\Delta t \to 0} \frac{\vec{r}(t + \Delta t) - \vec{r}(t)}{\Delta t} = \lim_{\Delta t \to 0} \frac{\Delta \vec{r}}{\Delta t} = \frac{d\vec{r}}{dt}$$

Speed

The ***magnitude*** of the velocity vector is called the ***speed***, ***v***, defined as:

$$v(t) = |\vec{v}(t)|$$

Decreasing time intervals Δt for the motion of the cheetah



Time Derivatives of Vector Functions

$$\vec{\boldsymbol{v}}(t) = \frac{d}{dt}\vec{\boldsymbol{r}}(t) = \frac{d}{dt}(x(t)\,\boldsymbol{i} + y(t)\,\boldsymbol{j} + z(t)\,\boldsymbol{k}) = \frac{dx}{dt}\boldsymbol{i} + \frac{dy}{dt}\boldsymbol{j} + \frac{dz}{dt}\boldsymbol{k}$$

$$v_x(t) = \frac{dx}{dt}, v_y(t) = \frac{dy}{dt}, v_z(t) = \frac{dz}{dt}$$

The velocity vector can therefore also be written:

$$\vec{\boldsymbol{v}}(t) = v_x(t)\,\boldsymbol{i} + v_y(t)\,\boldsymbol{j} + v_z(t)\,\boldsymbol{k} = \big(v_x(t), v_y(t), v_z(t)\big)$$

The **average acceleration** over a time interval Δt from t to t + Δt is:

$$\hat{a}(t_i) = \frac{\vec{v}(t+\Delta t) - \vec{v}(t)}{\Delta t} = \frac{\Delta \vec{v}(t_1)}{\Delta t}$$

We define the **instantaneous acceleration vector**, or simply the **instantaneous acceleration**, as the limit of the average acceleration vector when the time interval approaches zero:

$$\vec{a}(t_i) = \lim_{\Delta t \to 0} \frac{\vec{v}(t+\Delta t) - \vec{v}(t)}{\Delta t} = \frac{d\vec{v}}{dt} = \dot{v}$$

The acceleration vector is the time derivative of the velocity vector.

We find the acceleration in the vector component representation by componentwise derivation:

$$\vec{\boldsymbol{a}}(t) = \frac{d}{dt}\vec{\boldsymbol{v}}(\boldsymbol{t}) = \frac{d}{dt}\left(v_x(t)\,\boldsymbol{i} + v_y(t)\,\boldsymbol{j} + v_z(t)\,\boldsymbol{k}\right) = \frac{dv}{dt}\boldsymbol{i} + \frac{dv}{dt}\boldsymbol{j} + \frac{dv}{dt}\boldsymbol{k}$$

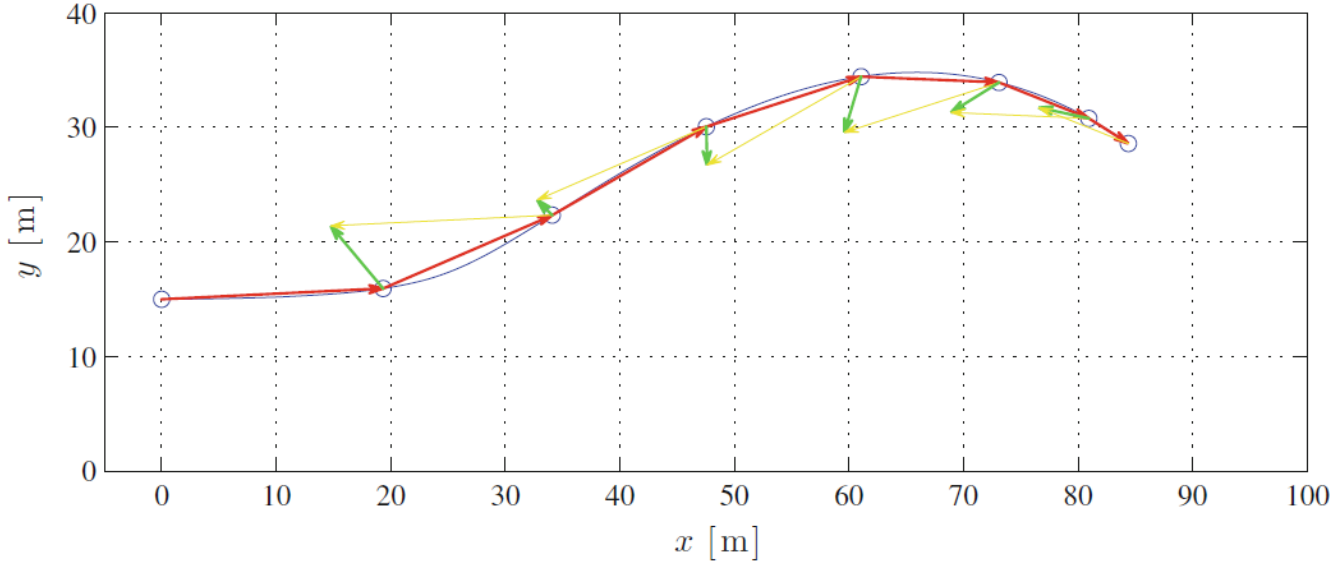
Since the velocity vector is the time derivative of the position vector

$$\vec{\boldsymbol{v}}(t) = \frac{d}{dt}\vec{\boldsymbol{r}} = \dot{\vec{\boldsymbol{r}}}$$

we can write the acceleration vector as the second time derivative of the position vector:

$$\vec{\boldsymbol{a}}(t) = \frac{d}{dt}\vec{\boldsymbol{v}} = \frac{d}{dt}\frac{d}{dt}\vec{\boldsymbol{r}} = \frac{d^2\vec{\boldsymbol{r}}}{dt^2} = \ddot{\vec{\boldsymbol{r}}}$$

Motion diagrams for the cheetah with $\Delta t = 0.5$ s illustrating both the displacements, interpreted as velocities, and the change in displacements, interpreted as accelerations. The constructions of the accelerations are illustrated.



```
import matplotlib.pyplot as plt
import numpy as np
t,x,y = np.loadtxt('soccer.txt', usecols=[0,1,2], unpack=True)
n = len(t)
dt = t[1] - t[0]
r = np.zeros((n, 2), float)
r[:,0] = x
r[:,1] = y
```

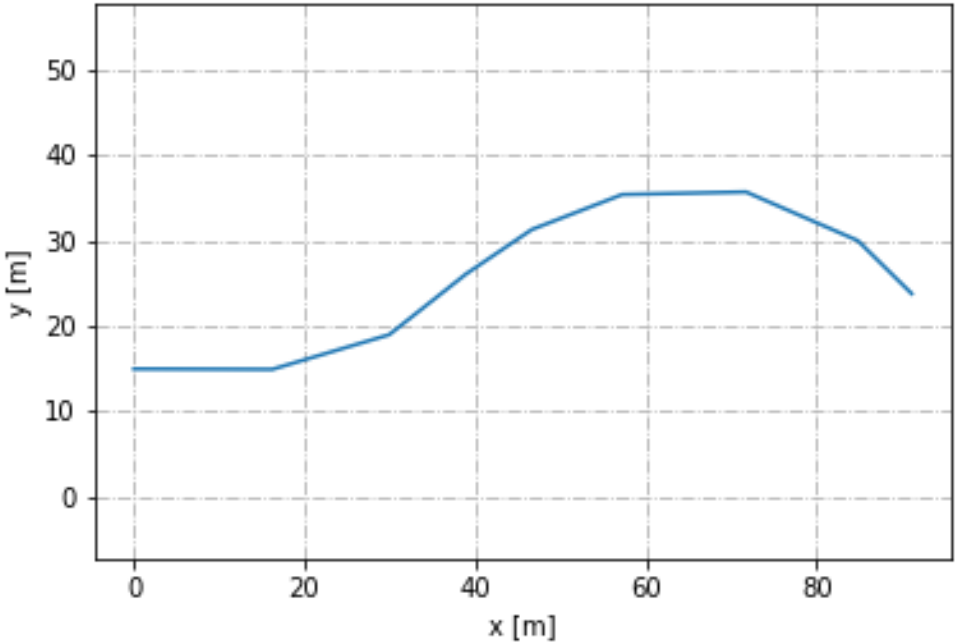
$$\vec{r}(t) = x(t)\boldsymbol{i} + y(t)\boldsymbol{j}$$

This generates the arrays *t*, *x*, and *y*, which we combine to one array to form r. This provides us with a vector $\boldsymbol{r}(t_i)$, which contains the x- and y-components

$$\vec{r}(t) = x(t)\boldsymbol{i} + y(t)\boldsymbol{j}$$

The vector representation in Python is useful and allows us to make operations on the whole vector in a very similar way to how we write the operations mathematically.

```
fig, ax = plt.subplots()
ax.plot(r[:,0],r[:,1])
ax.axis('equal')
ax.set_xlabel('x [m]')
ax.set_ylabel('y [m]')
plt.show()
```



where we have used **`axis('equal')`** to ensure that the scaling of the x- and y-axis are the same.

```
fig, ax = plt.subplots()
for i in range(n-1):
    ax.plot(r[i,0], r[i,1],'o')
    dr = r[i+1,:] - r[i,:]
    ax.quiver(r[i,0], r[i,1], dr[0], dr[1], angles='xy',
              scale_units='xy', scale=1)
ax.set_xlabel('x [m]')
ay.set_ylabel('y [m]')
plt.show()
```

quiver(**args*, ***kw*) Plot a 2-D field of arrows.

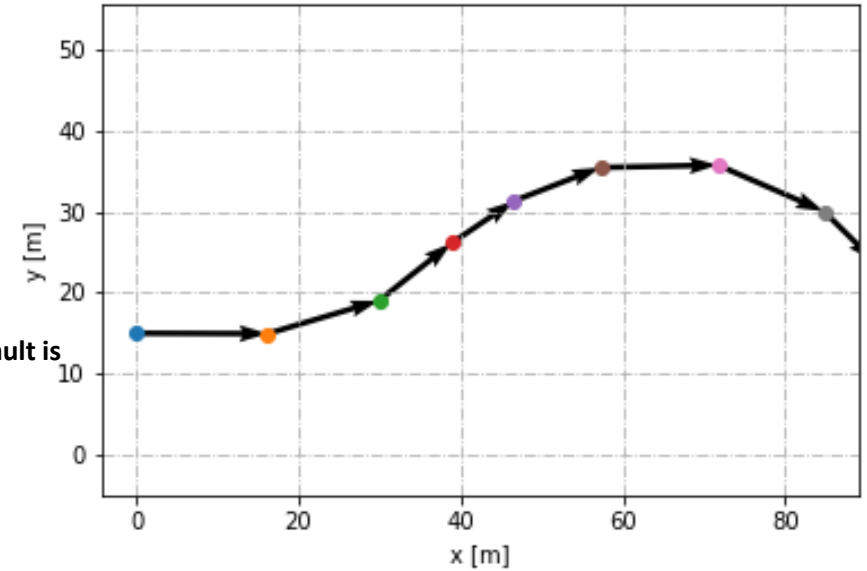
```
quiver(X, Y, U, V, C, **kw)
```

Arguments:

`X, Y`: The x and y coordinates of the arrow locations (default is tail of arrow; see pivot kwarg)

`U, V`: Give the x and y components of the arrow vectors

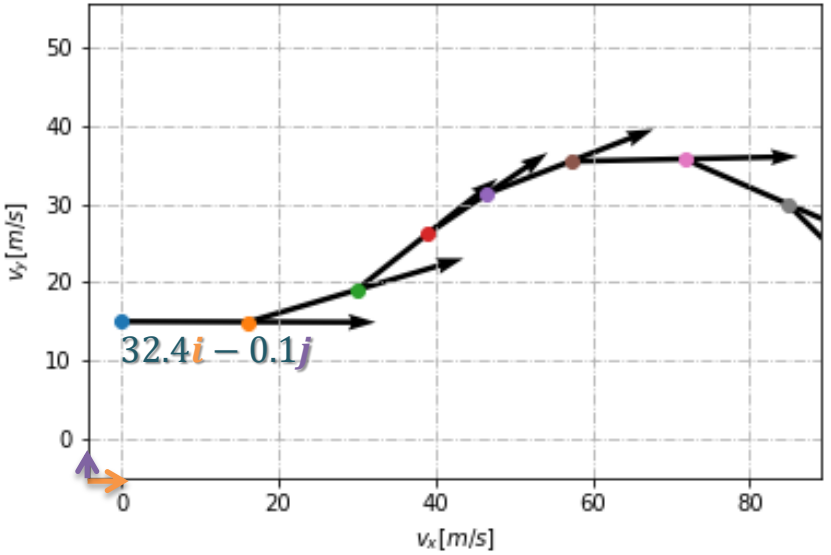
`C`: An optional array used to map colors to the arrows



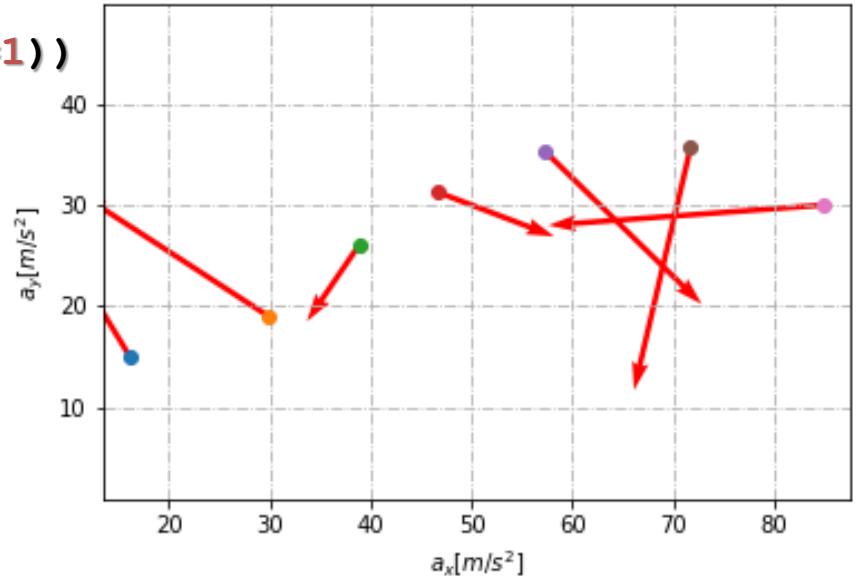
```
fig, ax = plt.subplots()
v = np.zeros((n-1, 2),float)
for i in range(n-1):
    ax.plot(r[i,0], r[i,1],'o')
    v[i,:] = (r[i+1,:] - r[i,:])/dt
    ax.quiver(r[i,0], r[i,1], v[i,0], v[i,1], angles='xy',
              scale_units='xy', scale=1)
ax.set_xlabel('vx [m/s]')
ax.set_ylabel('vy [m/s]')
plt.show()
```

t_i (s)	0.0	0.5
x_i (m)	0.0	16.2
y_i (m)	15.0	14.95

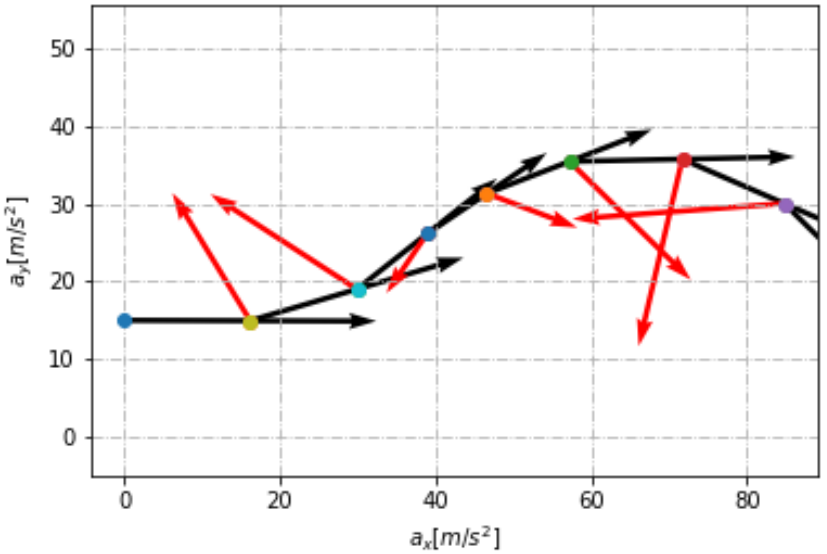
$$\vec{v}(0) = \frac{\Delta\vec{r}}{\Delta t} == \frac{16.2\,\boldsymbol{i} - 0.05\,\boldsymbol{j}}{0.5} = 32.4\frac{m}{s}\boldsymbol{i} - 0.1\frac{m}{s}\boldsymbol{j}$$

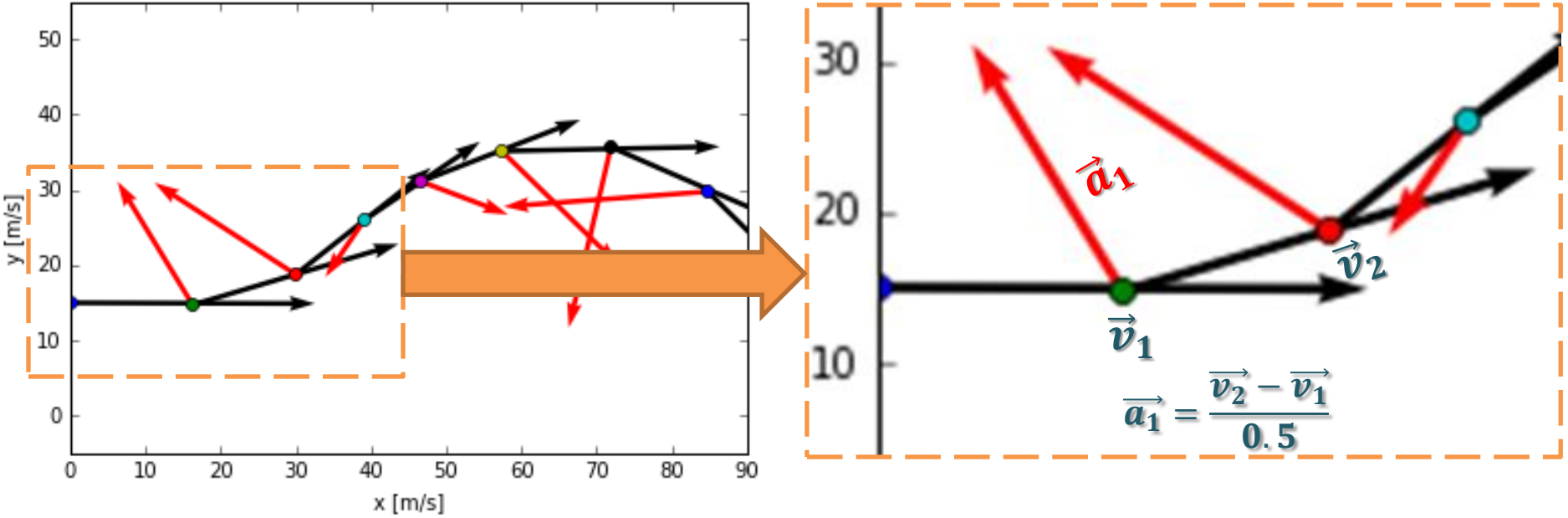


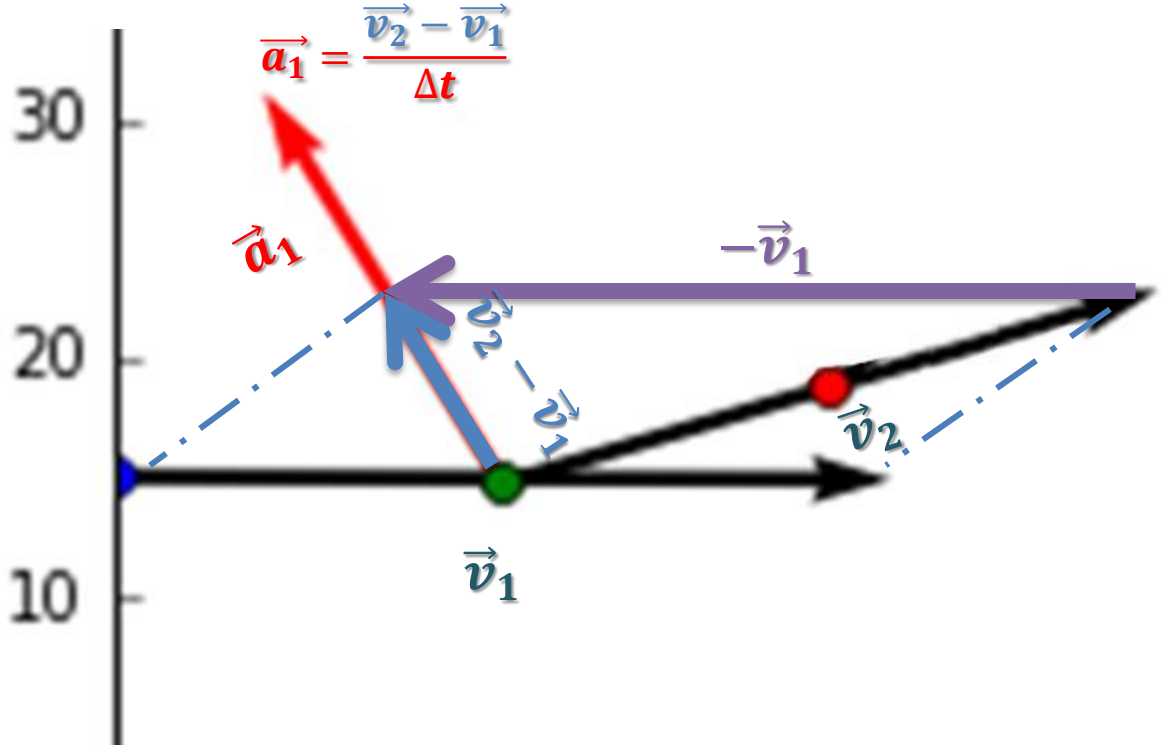
```
fig, ax = plt.subplots()
a = zeros((n-2, 2), float)
for i in range(n-2):
    ax.plot(r[i+1,0], r[i+1,1],'o')
    a[i,:] = (v[i+1,:] - v[i,:])/dt
    ax.quiver(r[i+1,0],r[i+1,1],a[i,0],a[i,1],\
    color='r',angles='xy',\
    scale_units='xy', scale=1))
ax.set_xlabel('ax [m/s/s]')
ax.set_ylabel('ay [m/s/s]')
plt.show()
```



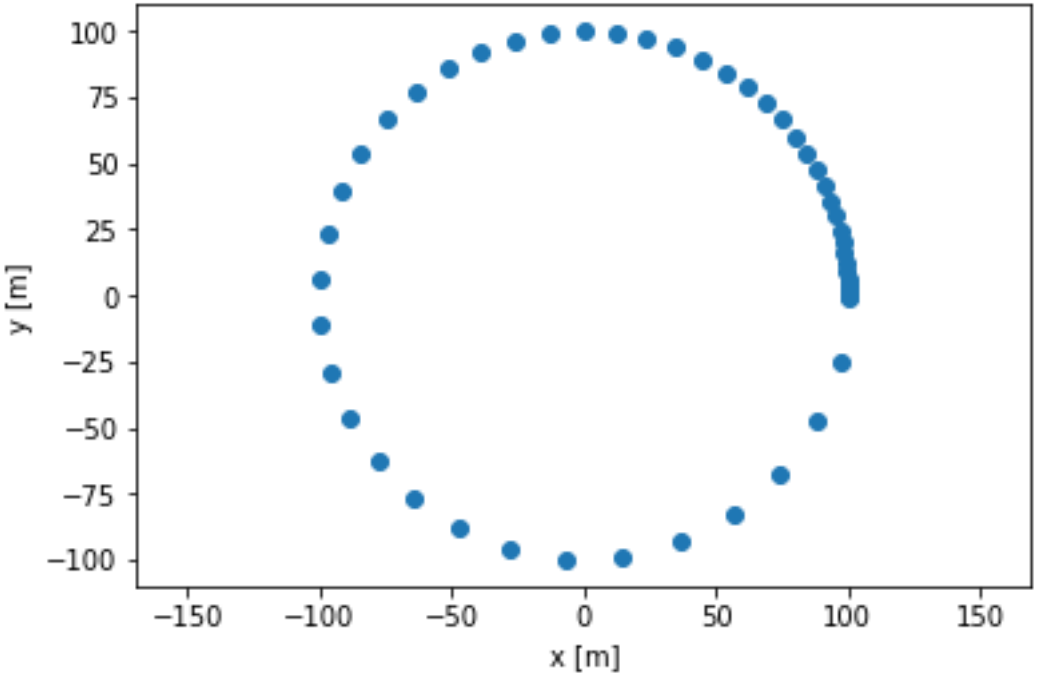
```
v = zeros((n-1, 2), float)
for i in range(n-1):
    plot(r[i,0], r[i,1],'o')
    v[i,:] = (r[i+1,:] - r[i,:])/dt
    quiver(r[i,0], r[i,1], v[i,0], v[i,1], angles = 'xy', scale_units = 'xy', scale = 1)
xlabel('vx [m/s]')
ylabel('vy [m/s]')
# NO SHOW()
a = zeros((n-2, 2), float)
for i in range(n-2):
    plot(r[i+1,0], r[i+1,1],'o')
    a[i,:] = (v[i+1,:] - v[i,:])/dt
    quiver(r[i+1,0], r[i+1,1], a[i,0], a[i,1],
          color='r', angles='xy',
          scale_units='xy', scale=1))
show()
```



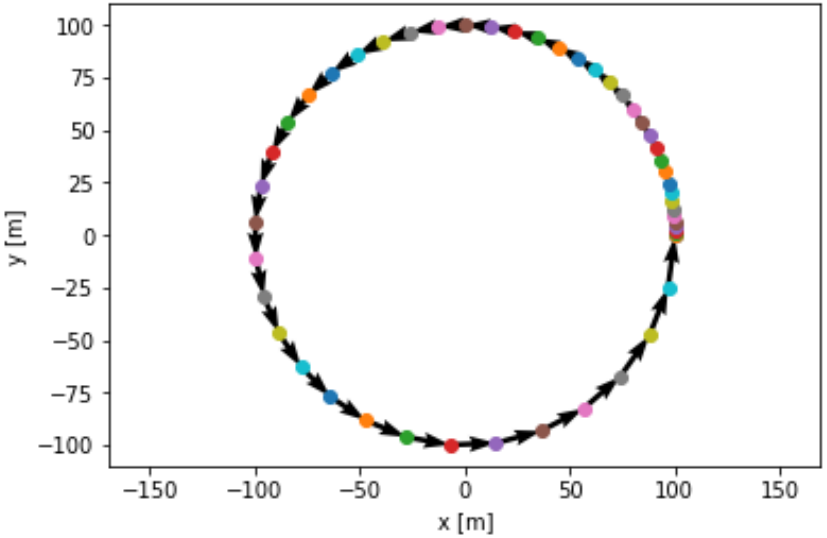

50
40
30
20
10
0
y [m/s]
0
10
20
30
40
50
60
70
80
90
x [m/s]
30
20
10
$\vec{a}_1$
$\vec{v}_2$
$\vec{v}_1$
$\overrightarrow{a_1} = \frac{\overrightarrow{v_2} - \overrightarrow{v_1}}{0.5}$

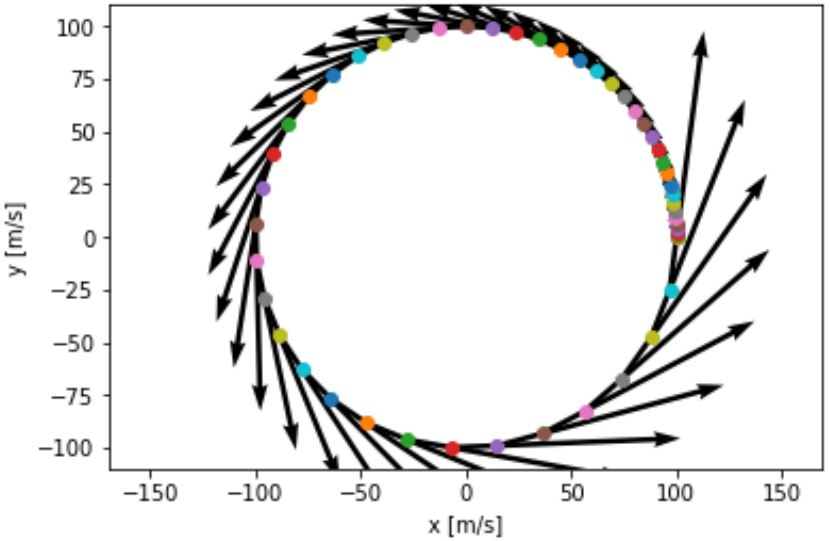

$\overrightarrow{a_1} = \frac{\overrightarrow{v_2} - \overrightarrow{v_1}}{\Delta t}$
30
20
10
$\vec{a}_1$
$-\vec{v}_1$
$\vec{v}_2 - \vec{v}_1$
$\vec{v}_2$
$\vec{v}_1$

Classwork

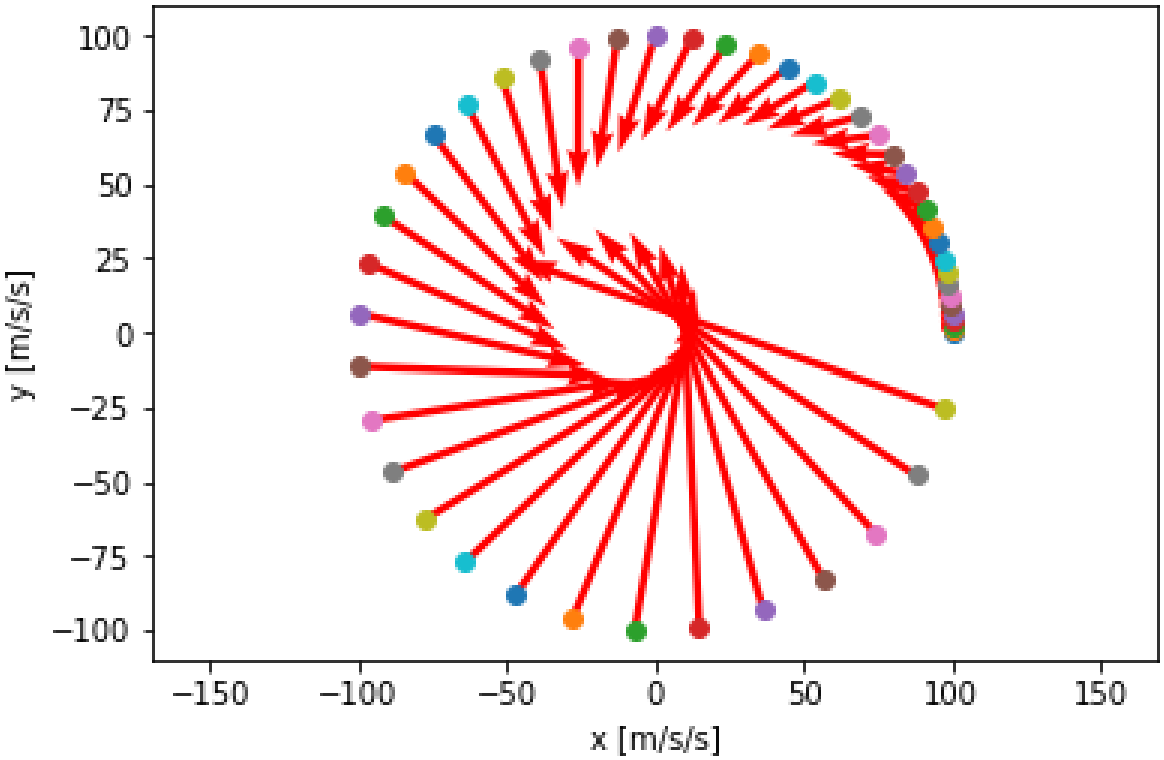


Classwork

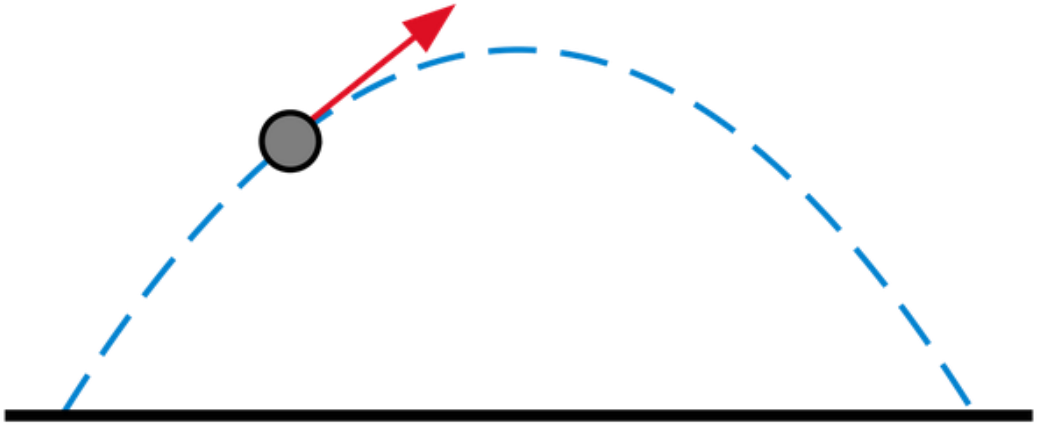

100
75
50
25
0
−25
−50
−75
−100
y [m]
−150
−100
−50
0
50
100
150
x [m]

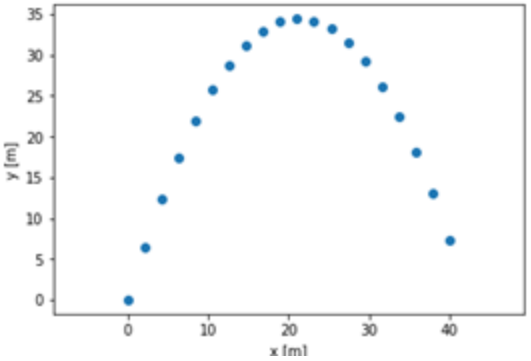

100
75
50
25
0
−25
−50
−75
−100
y [m/s]
−150
−100
−50
0
50
100
150
x [m/s]

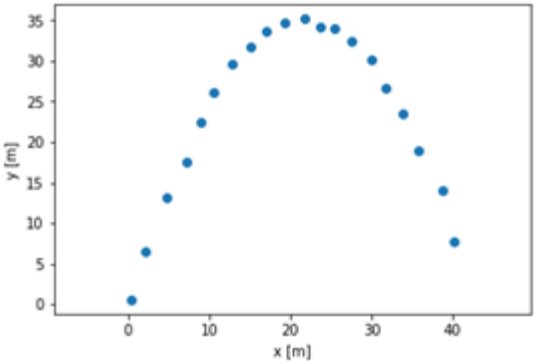
Classwork



Classwork







Classwork

projectile

X	Y
0.000	0.000
2.105	6.502
4.211	12.325
6.316	17.469
8.421	21.934
10.526	25.718
12.632	28.824
14.737	31.250
16.842	32.997
18.947	34.065
21.053	34.453
23.158	34.162
25.263	33.191
27.368	31.541
29.474	29.212
31.579	26.203
33.684	22.515
35.789	18.148
37.895	13.101
40.000	7.375

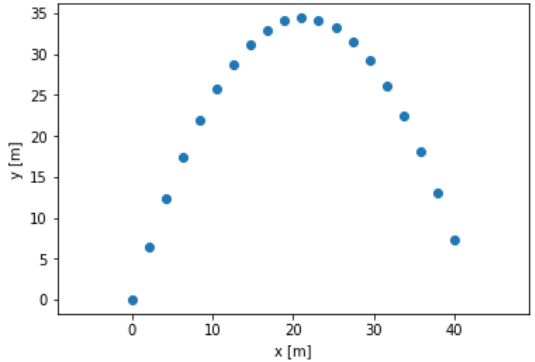
+

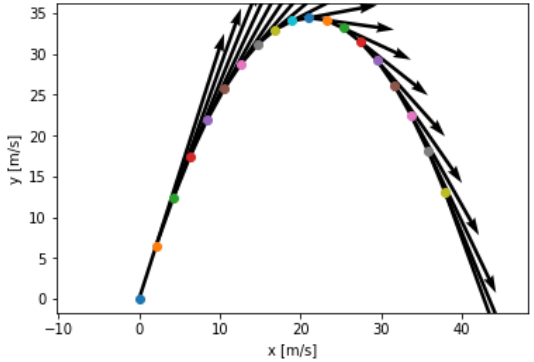
Noise	
0.002	0.022
0.050	0.061
0.067	0.002
0.001	0.049
0.006	0.098
0.065	0.045
0.045	0.013
0.023	0.047
0.051	0.037
0.070	0.099
0.059	0.066
0.068	0.059
0.020	0.072
0.047	0.089
0.012	0.035
0.042	0.023
0.035	0.067
0.072	0.008
0.095	0.077
0.065	0.021

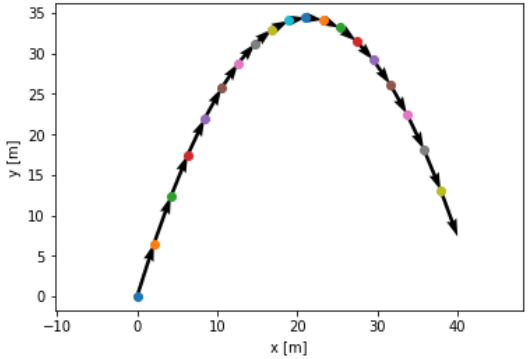
=

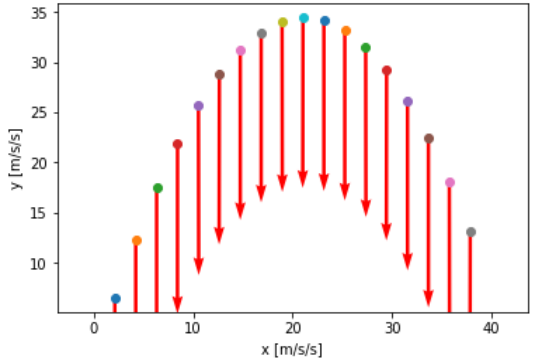
X	Y
0.002	0.022
2.155	6.564
4.278	12.327
6.317	17.518
8.427	22.032
10.591	25.763
12.677	28.837
14.760	31.297
16.893	33.034
19.017	34.164
21.112	34.519
23.226	34.221
25.283	33.263
27.415	31.630
29.486	29.247
31.621	26.226
33.719	22.582
35.861	18.156
37.990	13.178
40.065	7.396

Plot the displacement and the velocity values of tennis ball experiment

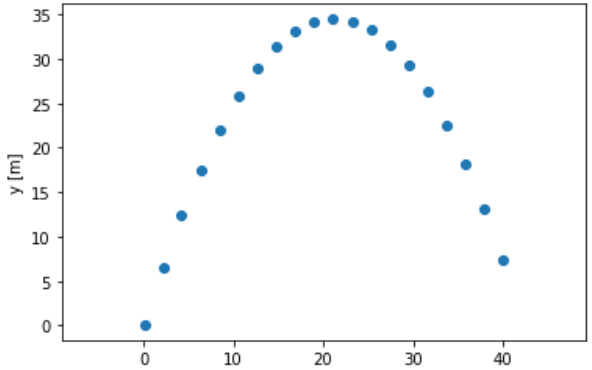


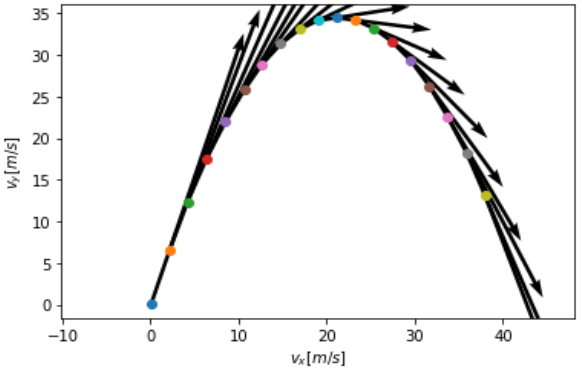


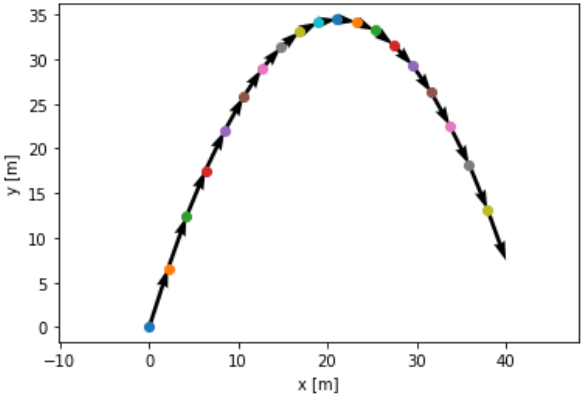


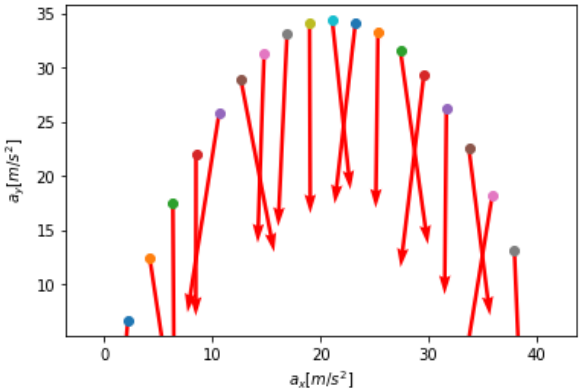


Plot the displacement and the velocity values of tennis ball experiment





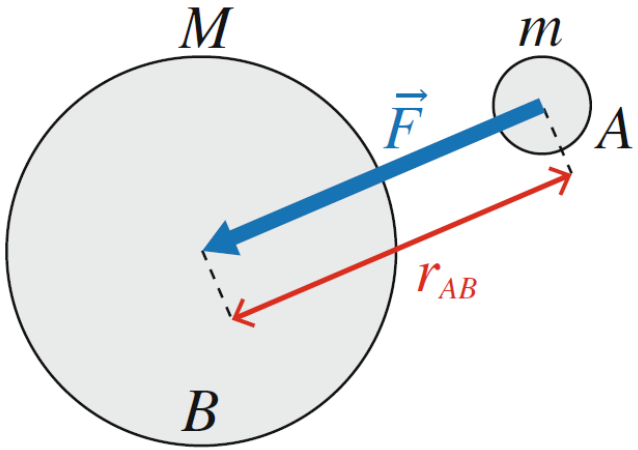




Gravitational Force

Another of Newton's great accomplishments is his discovery of the law of gravity. According to ***Newton's law of gravity***, there are attractive, gravitational forces between all objects. The gravitational force on object A from object B is:

$$\vec{F}_{from\ B\ on\ A} = G\frac{m.M}{r^2_{AB}}$$

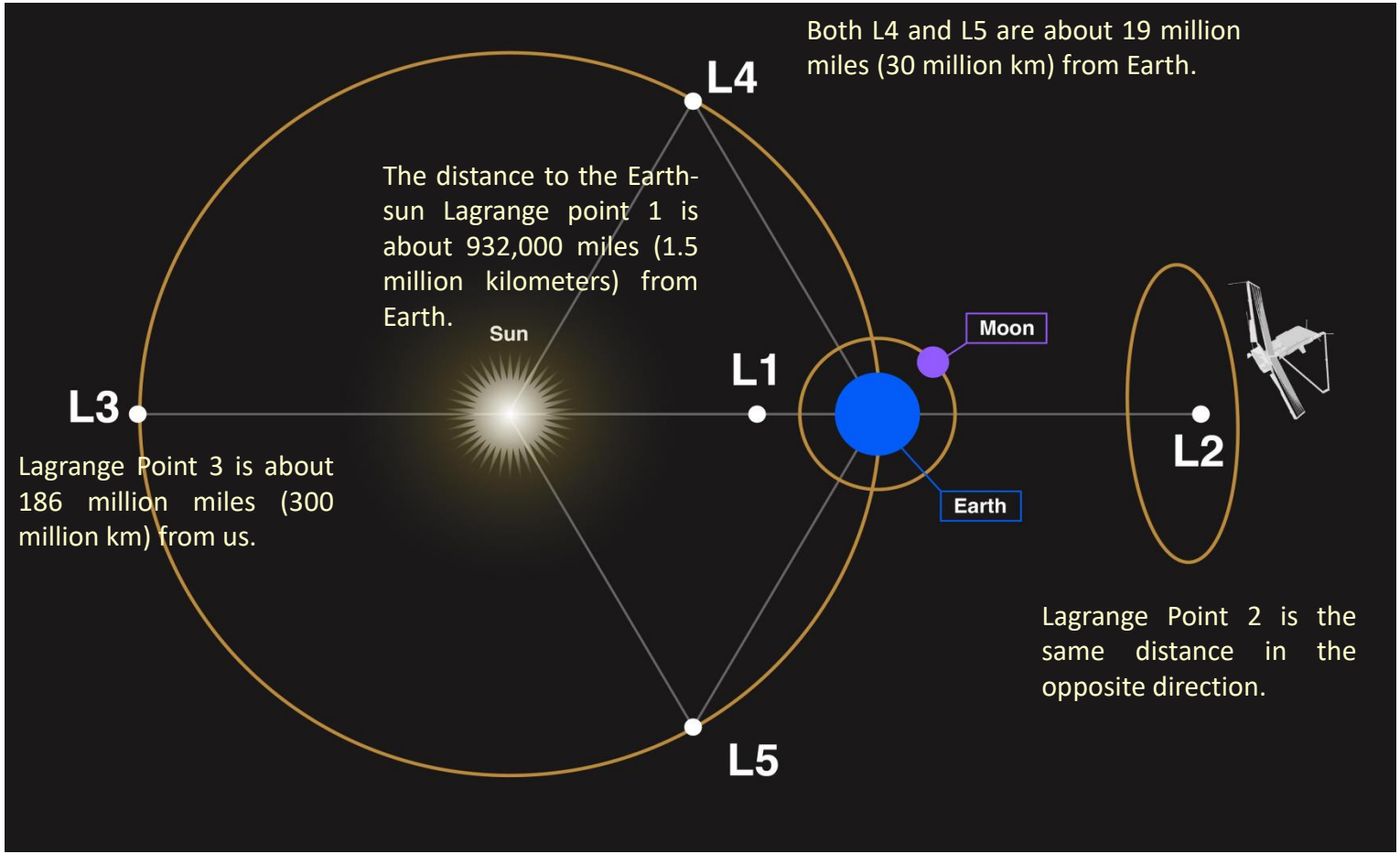


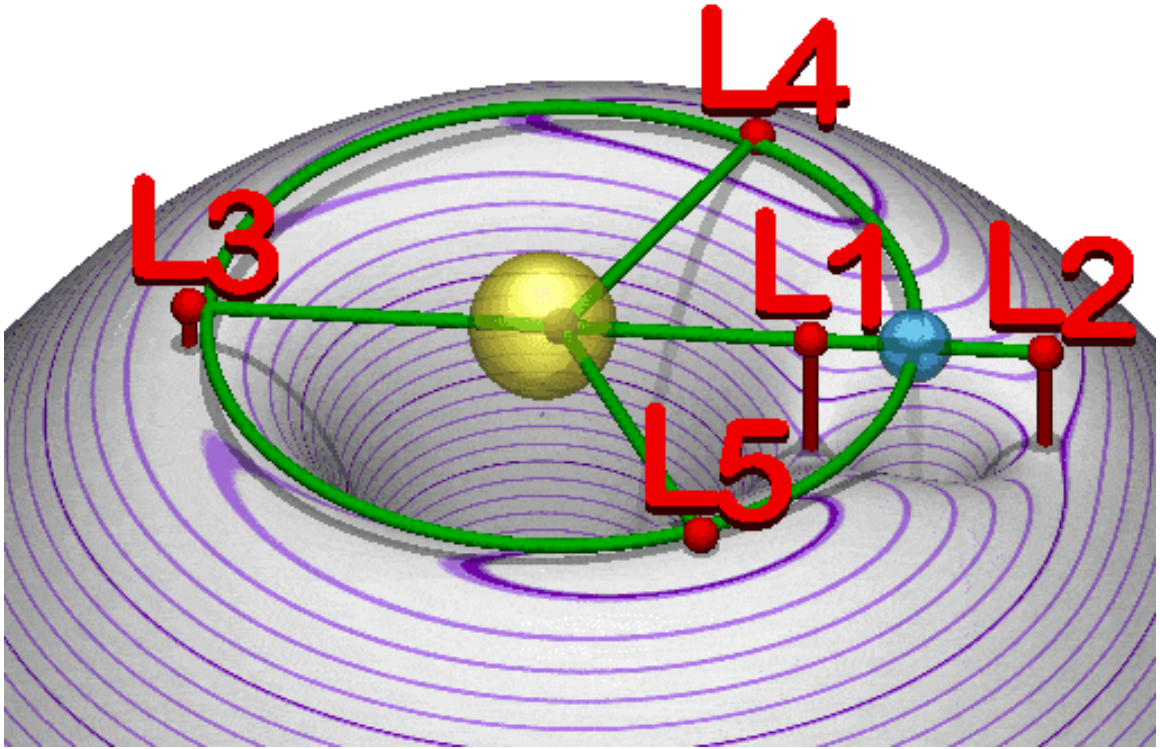
The force is **proportional** to the product of the two masses and inversely proportional to the **square of the distance** between them.

Constant Gravity

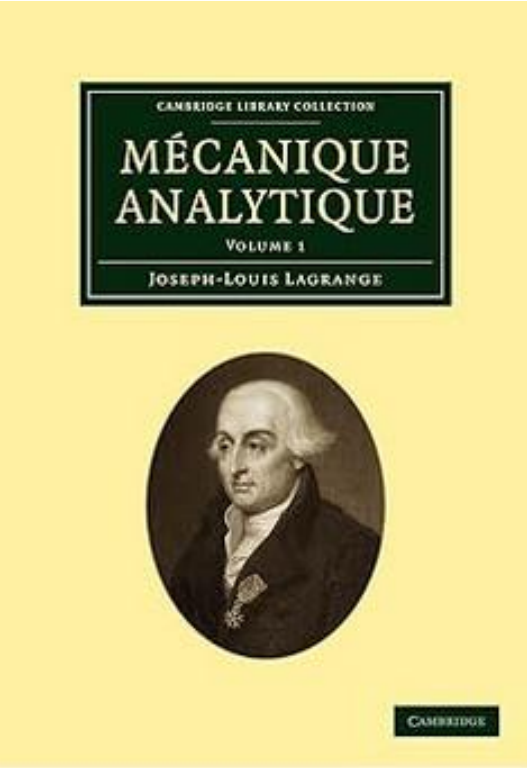
The acceleration of gravity on the surface of various objects in the Solar system.

Body	Mass (kg)	Radius (km)	g (m/s^2)	g/g_e
Sun	1.99×10^{30}	6.96×10^{5}	274.13	27.95
Mercury	3.18×10^{23}	2.43×10^{3}	3.59	0.37
Venus	4.88×10^{24}	6.06×10^{3}	8.87	0.90
Earth	5.98×10^{24}	6.38×10^{3}	9.81	1.00
Moon	7.36×10^{22}	1.74×10^{3}	1.62	0.17
Mars	6.42×10^{23}	3.37×10^{3}	3.77	0.38
Jupiter	1.90×10^{27}	6.99×10^{4}	25.95	2.65
Saturn	5.68×10^{26}	5.85×10^{4}	11.08	1.13
Uranus	8.68×10^{25}	2.33×10^{4}	10.67	1.09
Neptune	1.03×10^{26}	2.21×10^{4}	14.07	1.43
Pluto	1.40×10^{22}	1.50×10^{3}	0.42	0.04


Both L4 and L5 are about 19 million miles (30 million km) from Earth.
L4
The distance to the Earth-sun Lagrange point 1 is about 932,000 miles (1.5 million kilometers) from Earth.
Sun
Moon
L1
L3
L2
Earth
Lagrange Point 3 is about 186 million miles (300 million km) from us.
Lagrange Point 2 is the same distance in the opposite direction.
L5

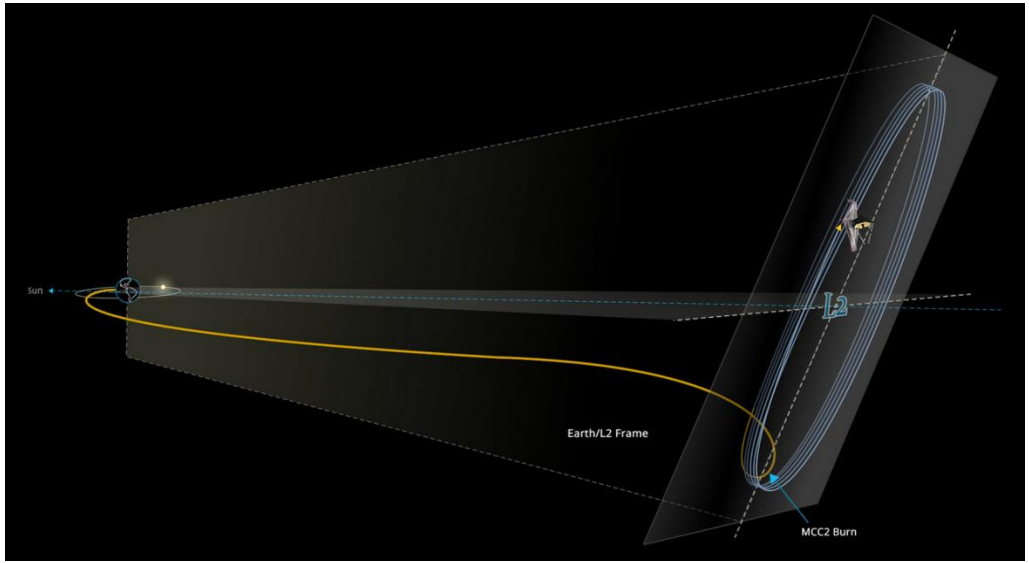

L4
L3
L1
L2
L5

Lagrange



Joseph-Louis Lagrange (born Giuseppe Lodovico [Luigi] Lagrangia, Turin, Piedmont, 25 January 1736 – Paris, 10 April 1813) was a mathematician and astronomer.

Lagrange's treatise on analytical mechanics, first published in 1788, was the best treatment of classical mechanics since Newton, and helped the development of mathematical physics in the nineteenth century.

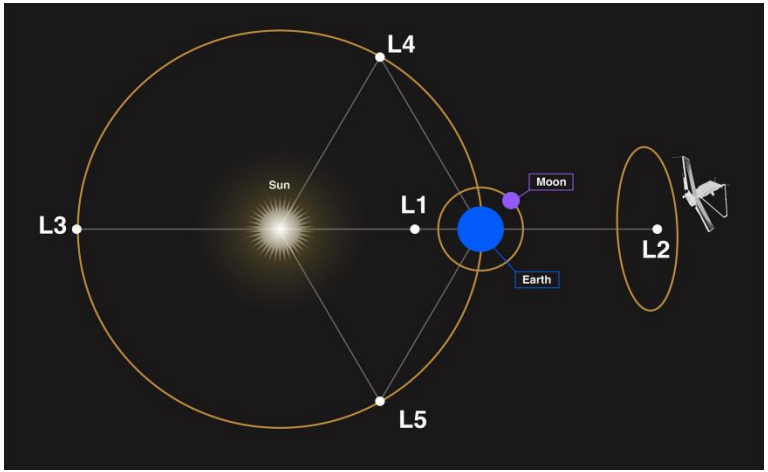


WEBB

Webb is not located at L2, but instead orbits L2, completing one circuit every 168 days. This "halo orbit" around L2 is highly elliptical and is roughly perpendicular to its orbital path around the Sun.

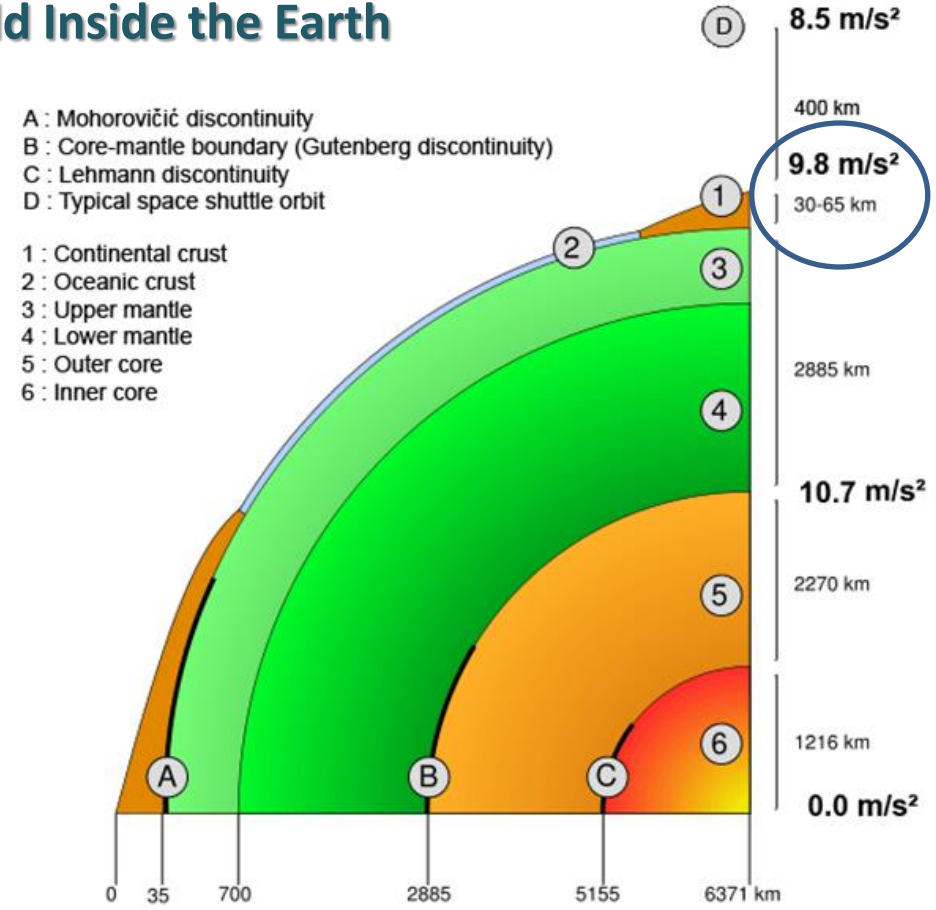
Diagram of the Lagrange Points associated with the Sun-Earth system. WEBB orbits around L2, which is about 1.5 million km from the Earth. Lagrange Points are positions in space where the gravitational forces of a two body system like the Sun and the Earth produce enhanced regions of attraction and repulsion. These can be used by spacecraft to reduce fuel consumption needed to remain in position.

PHYSICS in COMPUTER ANIMATIONS and GAMES

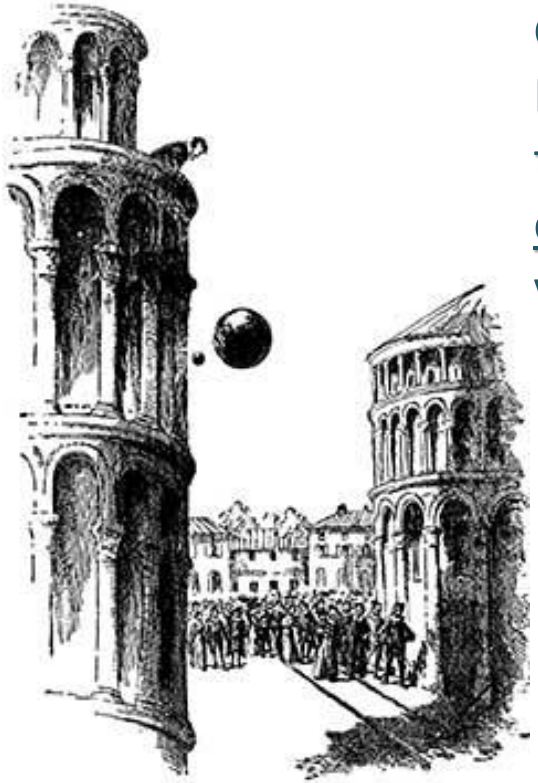


The James Webb Space Telescope orbits the Sun near Sun-Earth Lagrange point 2 (L2), approximately 1.5 million kilometers (1 million miles) from Earth. L2 is one of five Sun-Earth Lagrange points, positions in space where the gravitational pull of the Sun and Earth combine such that small objects in that region have the same orbital period (length of year) as Earth. This makes it possible for Webb to remain in constant communication with Earth.

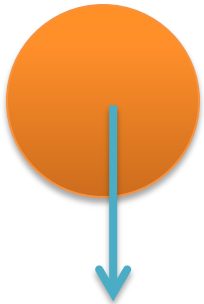
Gravitational Field Inside the Earth



Free Fall of an Object: An Experiment by Galileo

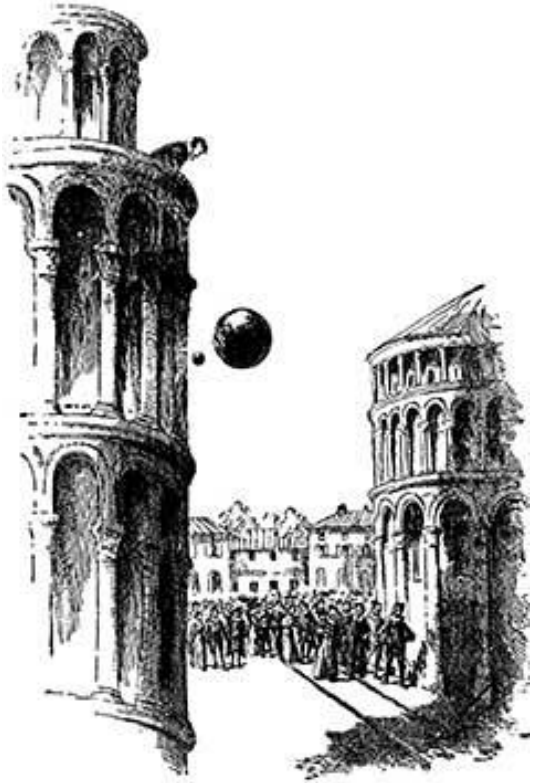


Galileo didn't know calculus (because Newton and Leibniz hadn't discovered it yet) so he couldn't derive the equation mathematically. Since we do know calculus (**SymPy**) we know that acceleration is the variation of velocity with time.



$$\vec{F}_{gravity} = m\vec{g}.$$

Free Fall of an Object: An Experiment by Galileo



We assume no drag/air resistance $\vec{F}_{gravity} = m\vec{g}$

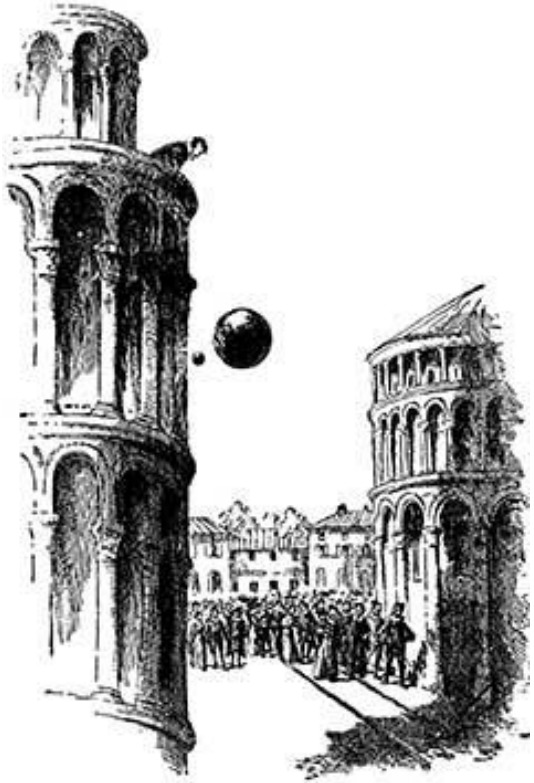
$$-\vec{g} = \frac{dv}{dt} = \frac{d^2x}{dt^2}$$

$$dv = -\vec{g}dt$$

$$v = \int -\vec{g}dt$$

$$v = -\vec{g}t$$

Free Fall of an Object: An Experiment by Galileo

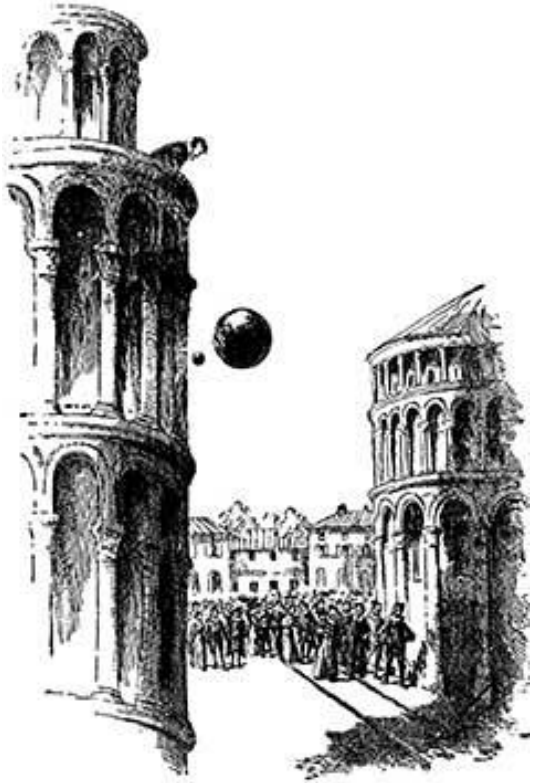


We assume **no** drag/air resistance

$$\int v dt = \int -\vec{g} t dt$$

$$x = -\frac{1}{2}\vec{g}t^2$$

Free Fall of an Object: Analytical Solution



We assume no drag/air resistance

$$v_t = v_0 - \vec{g}t$$

$$\frac{dx}{dt} = v_0 - \vec{g}t$$

$$\int \frac{dx}{dt} dt = \int [v_0 - \vec{g}t] dt$$

$$x_t - x_0 = v_0 t - \frac{1}{2}\vec{g}t^2$$

$$x_t = x_0 + v_0 t - \frac{1}{2}\vec{g}t^2$$

PHYSICS in COMPUTER ANIMATIONS and GAMES

Symbolic computation deals with the computation of mathematical objects symbolically. This means that the mathematical objects are represented exactly, not approximately, and mathematical expressions with unevaluated variables are left in symbolic form.

```
>>> import math
>>> math.sqrt(9)
>>> 3
```

9 is a perfect square, so we got the exact answer, 3. But suppose we computed the square root of a number that isn't a perfect square.

```
>>> math.sqrt(8)
>>> 2.82842712475
```

Here we got an approximate result. 2.82842712475 is not the exact square root of 8 (indeed, the actual square root of 8 cannot be represented by a finite decimal, since it is an irrational number).

suppose we want to go further. Recall that √8 = √4·2 = 2 √2. We would have a hard time deducing this from the above result. This is where symbolic computation comes in. With a symbolic computation system like SymPy, square roots of numbers that are not perfect squares are left unevaluated by default

```
>>> import sympy
>>> sympy.sqrt(9)
>>> 3
>>> symp.sqrt(8)
>>> 2*sqrt(2)
```

PHYSICS in COMPUTER ANIMATIONS and GAMES

The real power of a symbolic computation system such as SymPy is the ability to do all sorts of computations symbolically. SymPy can simplify expressions, compute derivatives, integrals, and limits, solve equations, work with matrices, and much, much more, and do it all symbolically.

```
>>> from sympy import *
>>> x, t, z, nu = symbols('x t z nu')
>>> init_printing(use_unicode=True)
>>> diff(sin(x)*exp(x), x)
 x           x
e ·sin(x) + e ·cos(x)
>>> integrate(exp(x)*sin(x) + exp(x)*cos(x), x)
 x
e ·sin(x)
```

Take the derivative of $\sin(x)e^x$.

PHYSICS in COMPUTER ANIMATIONS and GAMES

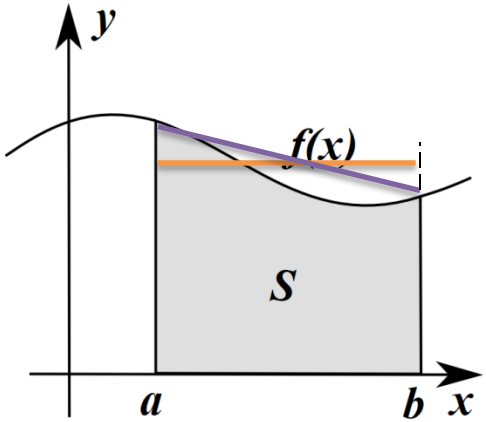

```
from sympy import *
v, x, t, g = symbols('v x t g', real = True)
m= symbols('m', constant = True)
init_printing(use_unicode = True)

v = integrate(-m*g, t)
print('Velocity : ', v)
x = integrate(v, t)
print('Position : ', x)

Velocity :  -g*m*t
Position :  -g*m*t**2/2
```

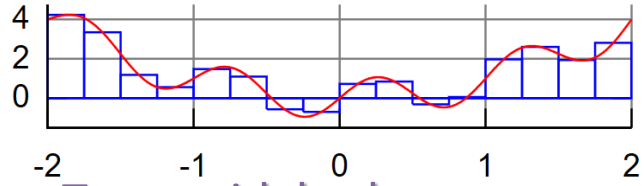
Note that SymPy does not include the constant of integration like mass [m]

Numerical integration



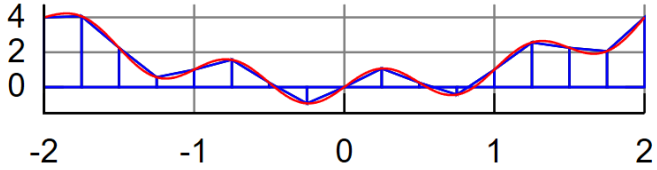
$$\int_a^b f(x)dx$$

Rectangle rule



$$\int_a^b f(x)dx \approx (b-a) f\left(\frac{a+b}{2}\right)$$

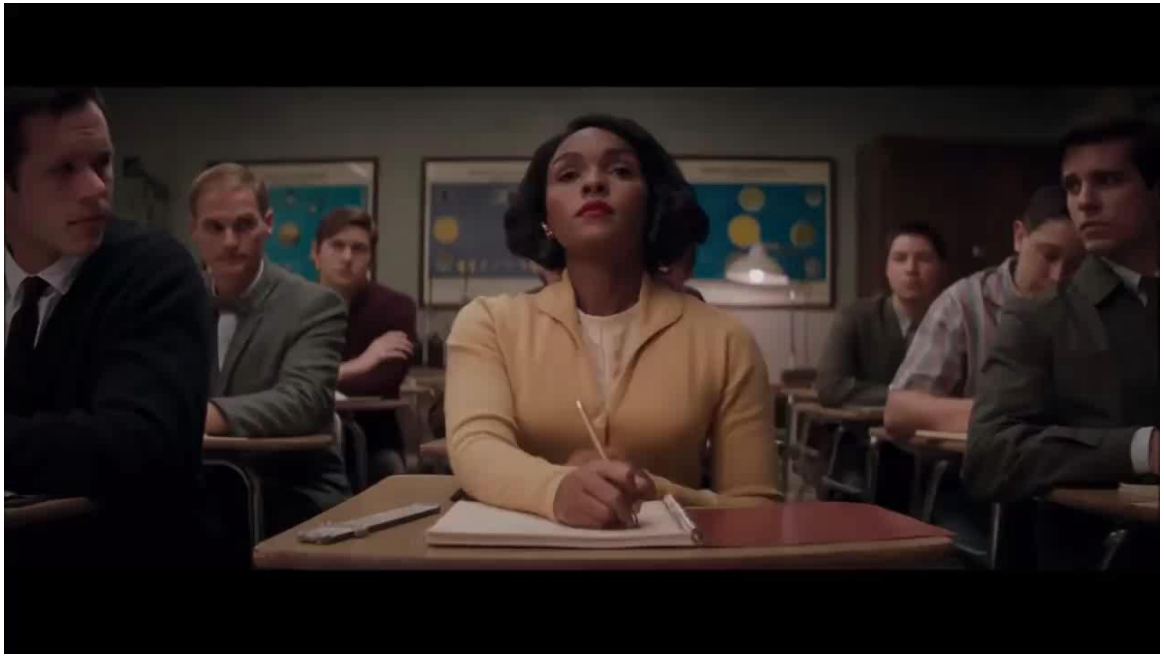
Trapezoidal rule



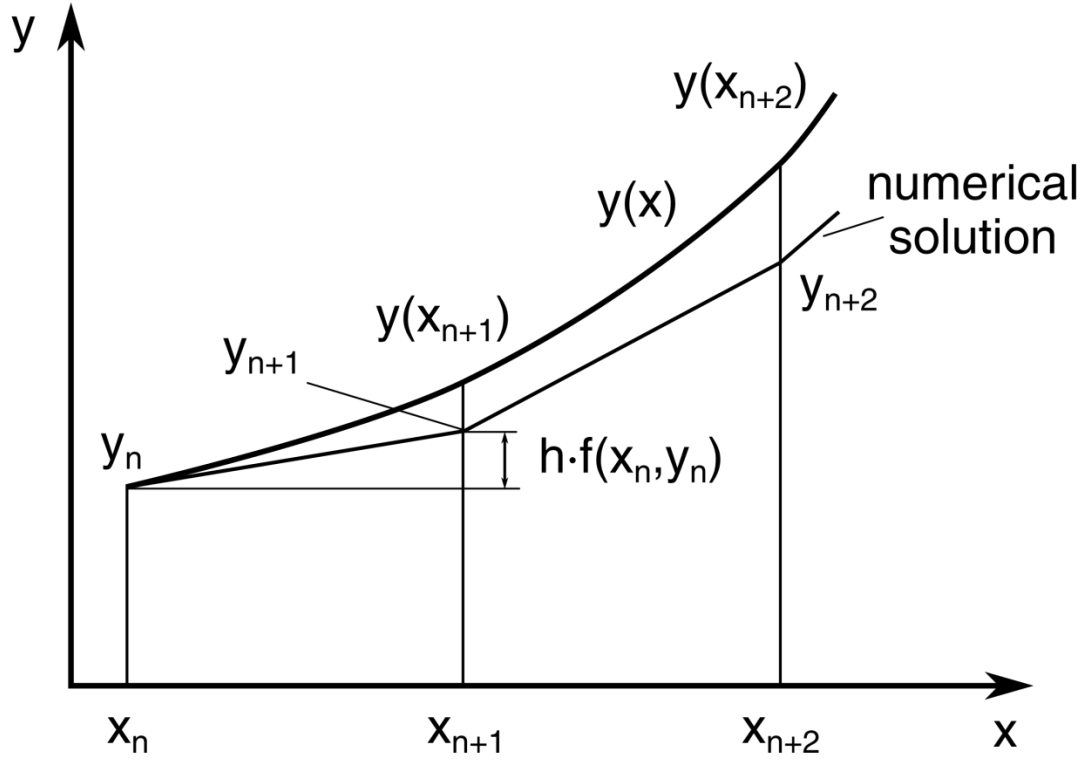
$$\int_a^b f(x)dx \approx (b-a)\frac{f(a)+f(b)}{2}$$

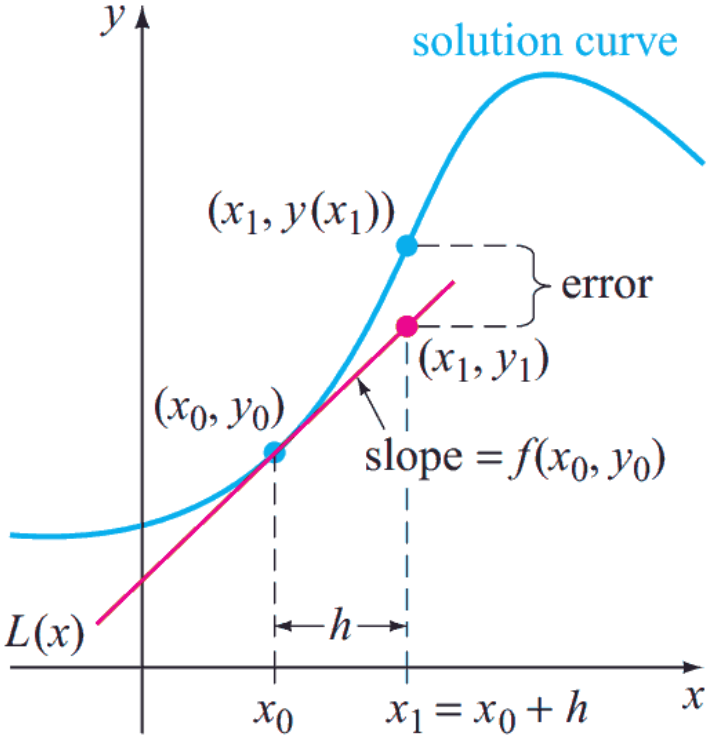
Euler method

Euler method : It’s ancient ….. but it's works

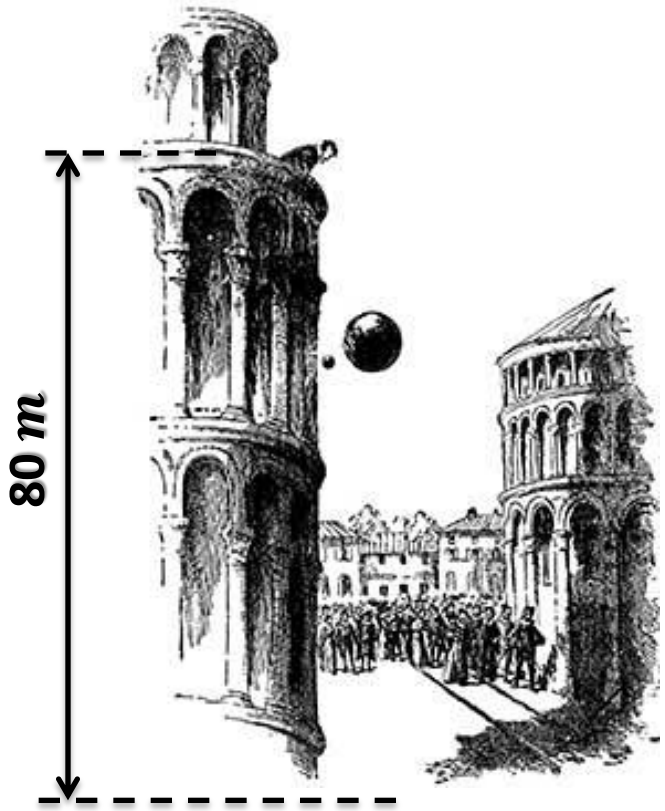


Euler method





Free Fall of an Object: Numerical Solution



The **Euler method** is a numerical procedure for solving ordinary differential equations (ODEs) with a given initial value. It is the most basic explicit method for numerical integration of ordinary differential equations and is the simplest **Runge–Kutta** method. The Euler method is named after **Leonhard Euler**, who treated it in his book ***Institutionum calculi integralis*** (published 1768–70).

x_0 is the height of the Pisa tower ~ 80 m
And initial velocity v_0 is zero.

Free Fall of an Object: Euler method

Given the initial value problem

$$\frac{dy}{dt} = y \qquad y(0) = 1$$

we would like to use the Euler method to approximate ***y(4)*** *u*sing step size equal to 1 (***h* = 1**) that we can consider as a time step.

The Euler method is $y_{n+1} = y_n + hf(t_n, y_n)$

$$f(t_0, y_0) = f(0, 1) = 1$$

$$hf(t_0, y_0) = hf(0, 1) = 1.1 = 1$$

Free Fall of an Object: Euler method

Since the step size is the change in ***t***, when we multiply the step size and the slope of the tangent, we get a change in ***y*** value. This value is then added to the initial ***y*** value to obtain the next value to be used for computations.

$$y_{n+1} = y_n + hf(t_n, y_n)$$

$$y_1 = y_0 + hf(y_0) = 1 + 1.1 = 2$$

$$y_2 = y_1 + hf(y_1) = 2 + 1.2 = 4$$

$$y_3 = y_2 + hf(y_2) = 4 + 1.3 = 8$$

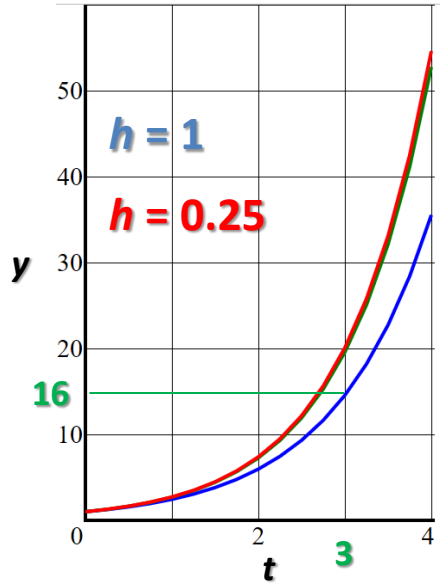
$$y_4 = y_3 + hf(y_3) = 4 + 1.4 = 16$$

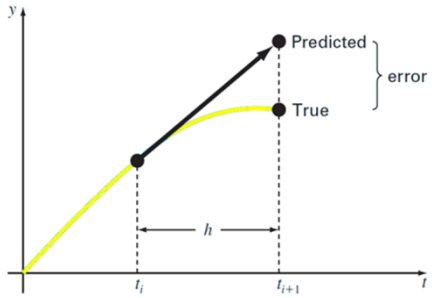
n	y_n	t_n	$f(t_n, y_n)$	h	Δy	y_{n+1}
0	1	0	1	1	1	2
1	2	1	2	1	2	4
2	4	2	4	1	4	8
3	8	3	8	1	8	16

Free Fall of an Object: Euler method

the Euler method is more accurate if the step size ***h*** is smaller. The table below shows the result with different step sizes. The top row corresponds to the example in the previous section, and the second row is illustrated in the figure.

step size	result of Euler's method	error
1	16	38.598
0.25	35.53	19.07
0.1	45.26	9.34
0.05	49.56	5.04
0.025	51.98	2.62
0.0125	53.26	1.34





Free Fall of an Object: Numerical Solution

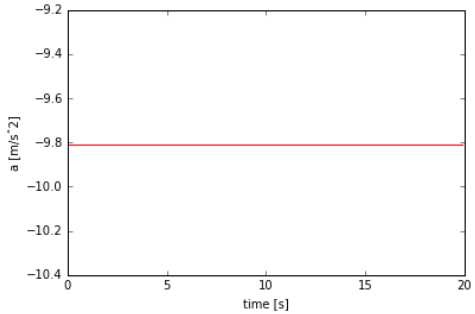
```
import numpy as np
import matplotlib.pyplot as plt
# Physical variables
g = 9.81          # gravity
time = 20.0       # Simulation Time
dt = 0.1          # (h) time step
# Numerical initialization
n = int(np.ceil(time/dt))
a = np.zeros(n,float)
v = np.zeros(n,float)
r = np.zeros(n,float)
t = np.zeros(n,float)
# Set initial values
r[0] = 80    # meters
v[0] = 0     # initial velocity
a[:] = -g    # constant acceleretion
# Integration loop
for i in range(n-1):
    v[i+1] = v[i] + a[i]*dt
    r[i+1] = r[i] + v[i+1]*dt
    t[i+1] = t[i] + dt
```

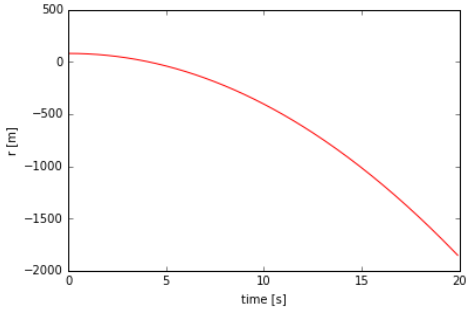
$$v(t_0 + \Delta t) \approx v(t_0) + a(t_0, r(t_0), v(t_0))\Delta t$$

$$r(t_0 + \Delta t) \approx r(t_0) + v(t_0 + \Delta t)\Delta t$$

Free Fall of an Object: Numerical Solution

```
# Acceleration Plotting
fig, ax = plt.subplots()
ax.plot(t, a,'-r')
ax.set_xlabel('time [s]')
ax.set_ylabel(r'$g [m/s^2]$')
plt.show()
# Velocity Plotting
fig, ax = plt.subplots()
ax.plot(t, v,'-r')
ax.set_xlabel('time [s]')
ax.set_ylabel(r'$v [m/s]$')
plt.show()
# Position Plotting
fig, ax = plt.subplots()
ax.plot(t, r, '-r')
ax.set_xlabel('time [s]')
ax.set_ylabel('r [m]')
plt.show()
```





Homework

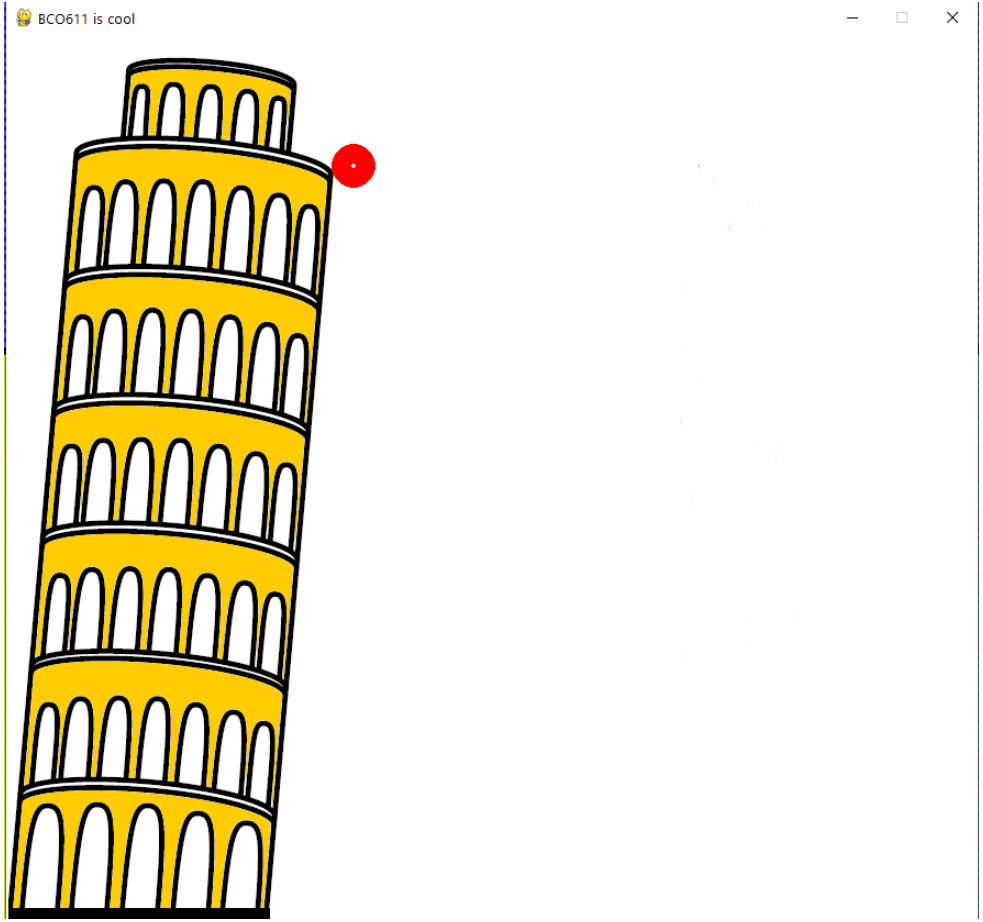
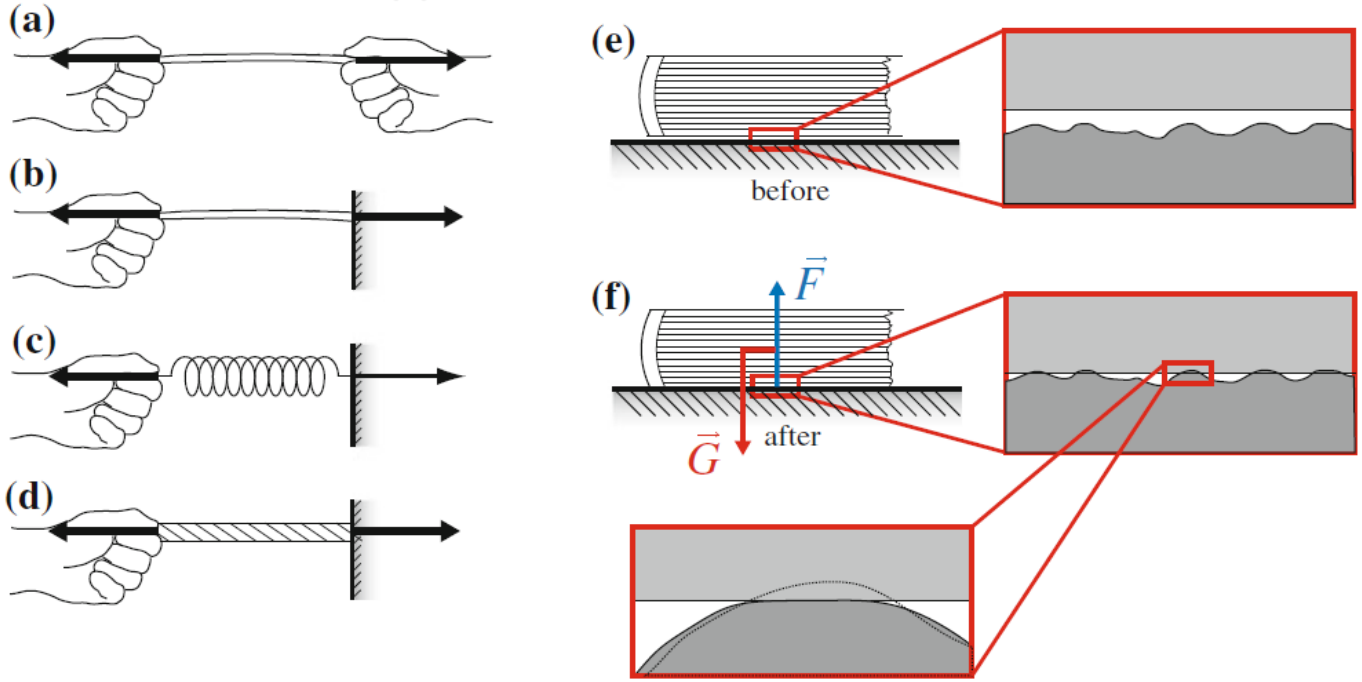


Illustration of (a) two hands pulling on a rubber band, (b) a rubber band attached to a wall, (c) a spring attached to a wall, (d) a rope attached to a wall, (e) a book above a table, (f) a book on a table



We could define a force as an interaction—a pull or a push on an object—that can be measured by the deformation of a spring. In this case the magnitude of the force increases with the deformation of the spring. This definition is not altogether satisfactory, but it illustrates a particular type of force— what we call a contact force. ***Contact forces*** occur where an object is in contact with other objects.

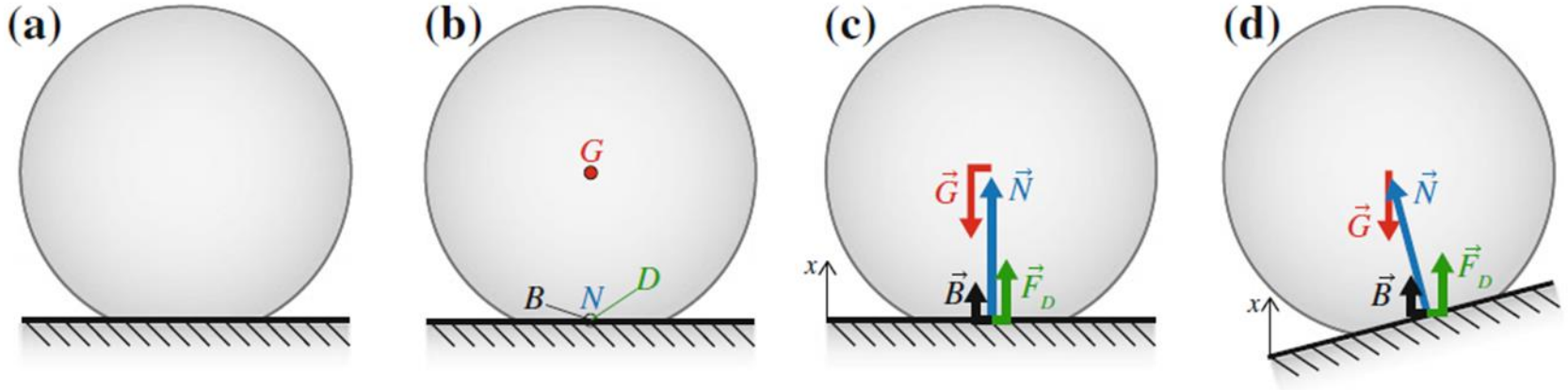
What about the book—where are the forces acting on the book? First, there is one force we have not discussed so far, the force of gravity. This is one of the fundamental forces in nature: There are ***gravitation forces*** between any two objects pulling the objects toward each other. There is a ***gravitational force*** from the Earth on the book, which pulls the book downward.

PHYSICS in COMPUTER ANIMATIONS and GAMES

If we zoom further in on the contact between one surface irregularity and the table, we realize that the contact force really is a sum of electromagnetic forces between the atoms on the surface of the book and the atoms on the surface of the table. The atoms are never in actual contact, but as the book and the table are pressed toward each other, electromagnetic forces will act from the table on the book. The electromagnetic force has been shown to be part of the electromagnetic and ***the weak nuclear force***, which is one of three fundamental forces. The other two are ***gravity*** and ***the strong nuclear force***, which is responsible for the interactions between subatomic particles and for the interactions in the nucleus. These are the three main forces in nature, and all forces are reducible to these forces. **In practice**, we cannot find the sum of the forces from all the individual atoms to find the magnitude of the force, but we will instead develop simplified models for the macroscopic forces we encounter.

Identifying Forces

First, we need to discern between the object, also called the system, and the environment, which is everything else. In this case, the system is the ball, and the environment is everything else, such as the floor, the air surrounding the ball, and the Earth.



Identifying Forces

Divide the problem into system and environment. In order to find the forces acting, we must realize a fundamental characteristic of a force:

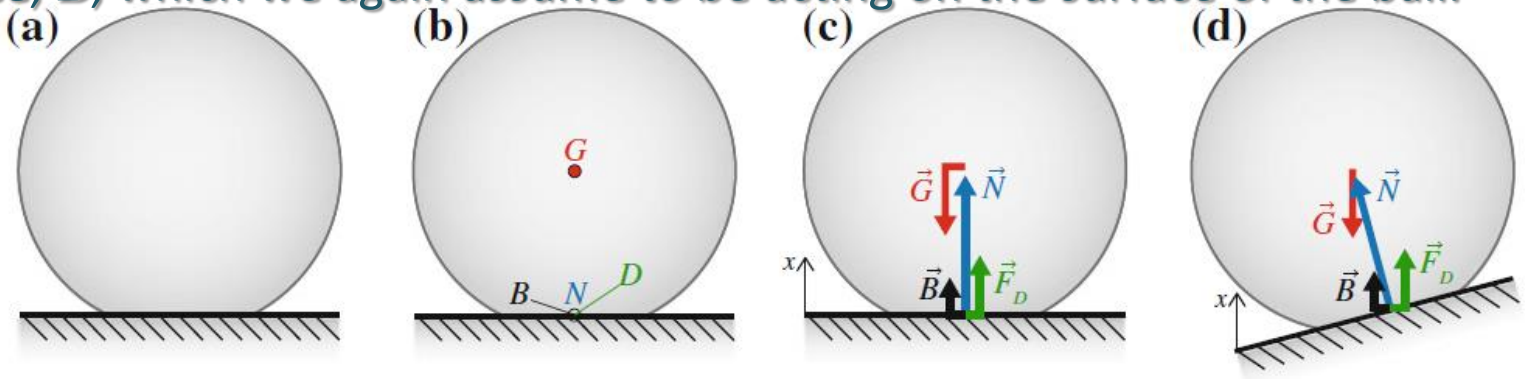
All forces acting on the system must have a source—an identifiable cause in the environment.

We have claimed that there are only three types of forces: gravity, the electromagnetic and weak nuclear force, and the strong nuclear force. However, this is not very helpful for our analysis of a macroscopic object such as the ball. Instead, we will divide forces into two main types:

Forces are either contact forces or long-range forces.

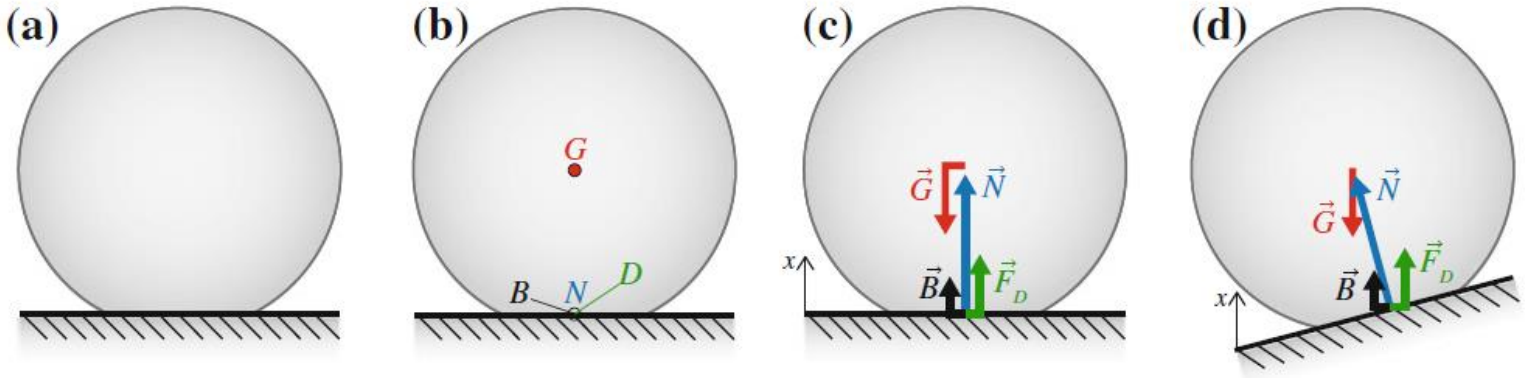
Identifying Forces

What is the contact force at this contact? The force on the ball is from the floor, and we call this force the normal force that must be a vector, and we introduce the symbol $\vec{\mathbf{N}}$ for the **normal force**. Again, we simplify by assuming that all these small forces sum to a single force, the **air resistance**, $\vec{\mathbf{F_D}}$, there are differences in the pressure in the air, which would give rise to a **buoyancy force**, $\vec{\mathbf{B}}$, which we again assume to be acting on the surface of the ball.



Identifying Forces

Finally, we must also look for the long-range forces affecting the ball. The only long-range force is the gravitational force acting from the Earth on the ball. We call this force, $\vec{G}$, and draw it as acting in the center of the ball, in the direction toward the center of the Earth.



Newton's Second Law of Motion

We are now able to find and identify the forces acting on an object. However, we still need a connection between the forces and the motion of an object. This connection can be found through Newton's second law of motion, which relates the acceleration of an object to the forces acting on the object:

Newton's second law of motion: The force $\vec{F}$ on an object of inertial mass m is related to the acceleration $\vec{a}$ of the object through $\vec{F} = m\vec{a}$.

Newton's second law is a vector equation: The acceleration is in the direction of the force, and the acceleration is proportional to the force. We determine the inertial mass of an object by measuring the acceleration for a given applied force.