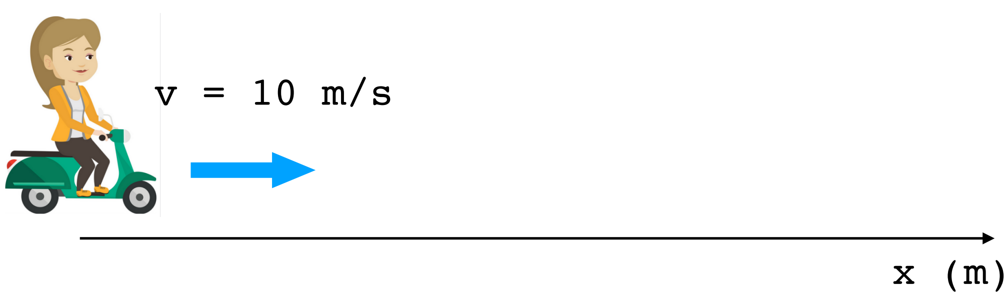
Modeling Constant-Velocity Motion

Track the Motion of a Scooter



We want to calculate the position of the scooter every 2 seconds

Given:
- velocity $v = 10$ m/s
- time step $\Delta t = 2\ s$
- initial position $x_0 = 0$
- initial time $t_0 = 0$
- final time $t_{max} = 10\ s$

Calculate the Position vs. Time:

Method 1: use constant velocity kinematic equation to find x(t) directly

$$x = x_0 + vt$$

$$t_n = 0, 2, 4, 6, 8, 10s$$
$$x_0 = 0\ m$$
$$v = 10\ m/s$$

snapshot (index)	time t_n	position x_n
n = 1	0 s	0 m
n = 2	2 s	20 m
n = 3	4 s	40 m
n = 4	6 s	60 m
n = 5	8 s	80 m
n = 6	10 s	100 m

Calculate the Position vs. Time:

Method 2: Calculate the position on step n+1 from position on step n

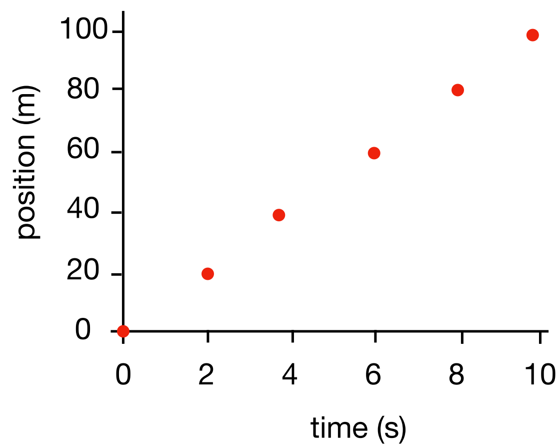
$$x_{n+1} = x_n + v\Delta t$$

$$v = 10\ m/s$$

Initial conditions (n = 1):	$t_1 = 0\ s$	$x_1 = 0\ m$
2nd step (n = 2):	$t_2 = t_1 + \Delta t$ $= 0 + 2$ $= 2\ s$	$x_2 = x_1 + v\Delta t$ $= 0 + (10\ m/s)(2\ s)$ $= 20\ m$
3rd step (n = 3):	$t_3 = t_2 + \Delta t$ $= 2 + 2$ $= 4\ s$	$x_3 = x_2 + v\Delta t$ $= 20 + (10\ m/s)(2\ s)$ $= 40\ m$

Position vs. Time Graph

snapshot (index)	time t_n	position x_n
n = 1	0 s	0 m
n = 2	2 s	20 m
n = 3	4 s	40 m
n = 4	6 s	60 m
n = 5	8 s	80 m
n = 6	10 s	100 m



Pseudocode

Initialization
1. Define: object velocity v
2. Define: time step and final time
3. Calculate number of points N
4. Preallocate arrays to store t and x values
5. Store initial conditions in x(1) and t(1)

Iteration
6. Loop to calculate t(n) and x(n) for n = 1 to N

Present Results
7. Plot x vs. t

Matlab Code

Initialization

1. Define: object velocity v

```
v = 10;
```

2. Define: time step and final time

```
dt = 2;
tmax = 10;
```

3. Calculate number of points N

```
N = ceil(tmax / dt + 1);
```

4. Preallocate arrays to store t and x values

```
t = zeros(1,N);
x = zeros(1,N);
```

Matlab Code

Initialization (cont.)

5. Store initial conditions in x(1) and t(1)

```
t(1) = 0;
x(1) = 0;
```

Iteration

6. Loop to calculate t(n) and x(n) for n = 1 to N

```
for n=1:N
    t(n+1) = t(n) + dt;
    x(n+1) = x(n) + v * dt;
end
```

Matlab Code

Present Results

7. Plot x vs. t

```
plot(t,x,'ro','MarkerFaceColor','r')
xlabel('t')
ylabel('x')
```

Review: Creating the Position Array

Preallocate the position array and fill with zeros:

```
x = zeros(1,N);
```

x(1)	x(2)	x(3)	x(4)	x(5)	x(6)
0	0	0	0	0	0

Set initial condition

```
x(1) = 0;
```

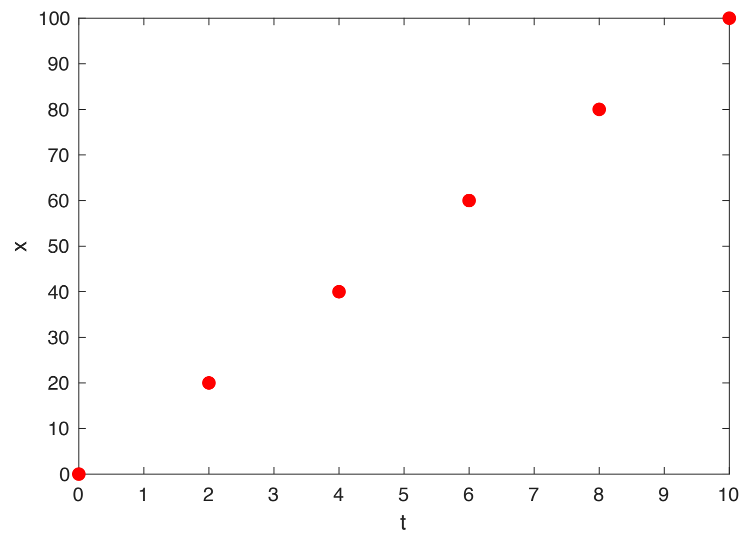
x(1)	x(2)	x(3)	x(4)	x(5)	x(6)
0	0	0	0	0	0

Loop fills in rest of values

x(1)	x(2)	x(3)	x(4)	x(5)	x(6)
0	20	40	60	80	100

```
for n=1:N
   x(n+1) = x(n) + v*dt;
end
```

Resulting Plot



Extend to Any First-Order ODE

$$\frac{dx}{dt} = f(x, t)$$

Modify the update rule for `x(n+1)`:

```
for n=1:N
    t(n+1) = t(n) + dt;
    x(n+1) = x(n) + v * dt;
end
```

replace with $f(x, t)$

Example

$$\frac{dx}{dt} = ax$$

Modify the update rule for `x(n+1)`:

```
for n=1:N
    t(n+1) = t(n) + dt;
    x(n+1) = x(n) + a * x(n) * dt;
end
```

Extend to a Second-Order ODE Derived from F=ma

$$F(x,t) = ma \quad \Rightarrow \quad \frac{dv}{dt} = a = \frac{1}{m} f(x,t), \qquad \frac{dx}{dt} = v$$

Modify the update rule to include velocity `v(n+1)`:

```
for n=1:N
    t(n+1) = t(n) + dt;
    x(n+1) = x(n) + v(n) * dt;
    v(n+1) = v(n) + a * dt;
end
```

replace with $\frac{f(x,t)}{m}$

Example

$$F(x,t) = -mg \quad \Rightarrow \quad \frac{dv}{dt} = a = -g, \quad \frac{dx}{dt} = v$$

Modify the update rule to include velocity `v(n+1)`:

```
for n=1:N
    t(n+1) = t(n) + dt;
    x(n+1) = x(n) + v(n) * dt;
    v(n+1) = v(n) - g * dt;
end
```