Critical Points and Optimization

We've explored various techniques that we can use to calculate the derivative of a function at a specific *x* value; in other words, we can determine the *slope* of the line created by the function at any point on the line.

This ability to calculate the slope means that we can use derivatives to determine some interesting properties of the function.

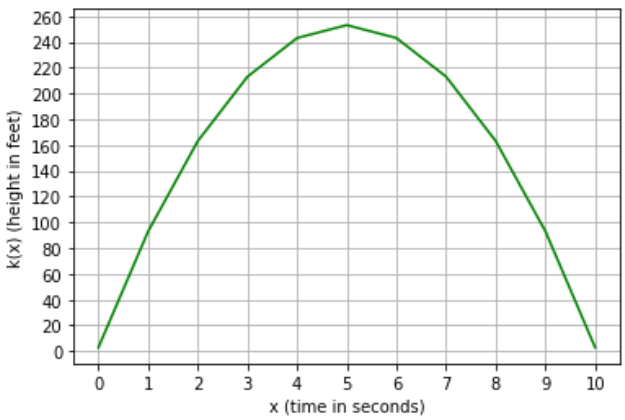
Function Direction at a Point

Consider the following function, which represents the trajectory of a ball that has been kicked on a football field:

$$k(x) = -10x^2 + 100x + 3$$

Run the Python code below to graph this function and see the trajectory of the ball over a period of 10 seconds.

In [1]:
```
%matplotlib inline

# Create function k
def k(x):
    return -10*(x**2) + (100*x)  + 3

from matplotlib import pyplot as plt

# Create an array of x values to plot
x = list(range(0, 11))

# Use the function to get the y values
y = [k(i) for i in x]

# Set up the graph
plt.xlabel('x (time in seconds)')
plt.ylabel('k(x) (height in feet)')
plt.xticks(range(0,15, 1))
plt.yticks(range(-200, 500, 20))
plt.grid()

# Plot the function
plt.plot(x,y, color='green')

plt.show()
```



By looking at the graph of this function, you can see that it describes a parabola in which the ball rose in height before falling back to the ground. On the graph, it's fairly easy to see when the ball was rising and when it was falling.

Of course, we can also use derivative to determine the slope of the function at any point. We can apply some of the rules we've discussed previously to determine the derivative function:

- We can add together the derivatives of the individual terms (***-10x^2***, ***100x***, and ***3***) to find the derivative of the entire function.
- The *power* rule tells us that the derivative of ***-10x^2*** is ***-10 • 2x***, which is ***-20x***.
- The *power* rule also tells us that the derivative of ***100x*** is ***100***.
- The derivative of any constant, such as ***3*** is ***0***.

So:

$$k'(x) = -20x + 100 + 0$$

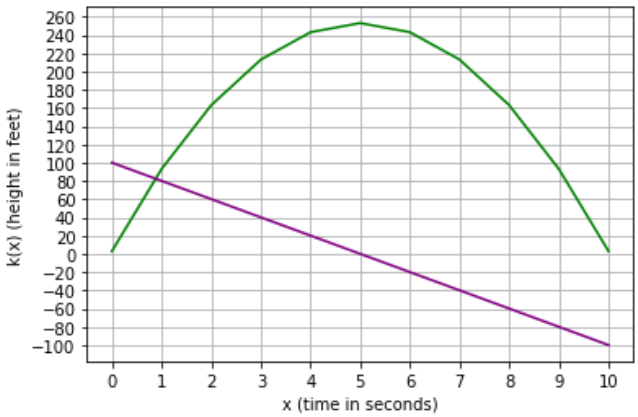
Which of course simplifies to:

$$k'(x) = -20x + 100$$

Now we can use this derivative function to find the slope for any value of ***x***.

Run the cell below to see a graph of the function and its derivative function:

In [2]:
```
%matplotlib inline

# Create function k
def k(x):
    return -10*(x**2) + (100*x)  + 3

def kd(x):
    return -20*x + 100

from matplotlib import pyplot as plt

# Create an array of x values to plot
x = list(range(0, 11))

# Use the function to get the y values
y = [k(i) for i in x]

# Use the derivative function to get the derivative values
yd = [kd(i) for i in x]

# Set up the graph
plt.xlabel('x (time in seconds)')
plt.ylabel('k(x) (height in feet)')
plt.xticks(range(0,15, 1))
plt.yticks(range(-200, 500, 20))
plt.grid()

# Plot the function
plt.plot(x,y, color='green')

# Plot the derivative
plt.plot(x,yd, color='purple')

plt.show()
```



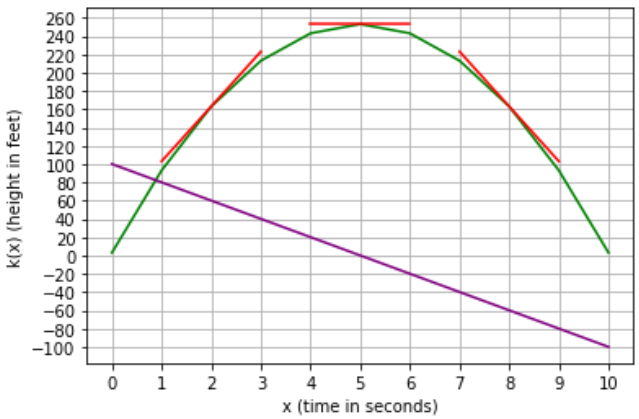
Look closely at the purple line representing the derivative function, and note that it is a constant decreasing value - in other words, the slope of the function is reducing linearly as x increases. Even though the function value itself is increasing for the first half of the parabola (while the ball is rising), the slope is becoming less steep (the ball is not rising at such a high rate), until finally the ball reaches its apogee and the slope becomes negative (the ball begins falling).

Note also that the point where the derivative line crosses 0 on the y-axis is also the point where the function value stops increasing and starts decreasing. When the slope has a positive value, the function is increasing; and when the slope has a negative value, the function is decreasing.

The fact that the derivative line crosses 0 at the highest point of the function makes sense if you think about it logically. If you were to draw the tangent line representing the slope at each point, it would be rotating clockwise throughout the graph, initially pointing up and to the right as the ball rises, and turning until it is pointing down and right as the ball falls. At the highest point, the tangent line would be perfectly horizontal, representing a slope of 0.

Run the following code to visualize this:

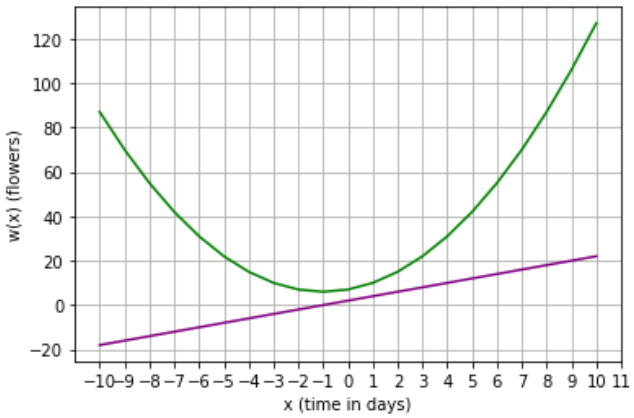
In [3]:
```
%matplotlib inline

# Create function k
def k(x):
    return -10*(x**2) + (100*x)  + 3

def kd(x):
    return -20*x + 100

from matplotlib import pyplot as plt

# Create an array of x values to plot
x = list(range(0, 11))

# Use the function to get the y values
y = [k(i) for i in x]

# Use the derivative function to get the derivative values
yd = [kd(i) for i in x]

# Set up the graph
plt.xlabel('x (time in seconds)')
plt.ylabel('k(x) (height in feet)')
plt.xticks(range(0,15, 1))
plt.yticks(range(-200, 500, 20))
plt.grid()

# Plot the function
plt.plot(x,y, color='green')

# Plot the derivative
plt.plot(x,yd, color='purple')

# Plot tangent slopes for x = 2, 5, and 8
x1 = 2
x2 = 5
x3 = 8
plt.plot([x1-1,x1+1],[k(x1)-(kd(x1)),k(x1)+(kd(x1))], color='red')
plt.plot([x2-1,x2+1],[k(x2)-(kd(x2)),k(x2)+(kd(x2))], color='red')
plt.plot([x3-1,x3+1],[k(x3)-(kd(x3)),k(x3)+(kd(x3))], color='red')

plt.show()
```



Now consider the following function, which represents the number of flowers growing in a flower bed before and after the spraying of a fertilizer:

$$w(x) = x^2 + 2x + 7$$

In [4]:
```
%matplotlib inline

# Create function w
def w(x):
    return (x**2) + (2*x) + 7

def wd(x):
    return 2*x + 2

from matplotlib import pyplot as plt

# Create an array of x values to plot
x = list(range(-10, 11))

# Use the function to get the y values
y = [w(i) for i in x]

# Use the derivative function to get the derivative values
yd = [wd(i) for i in x]

# Set up the graph
plt.xlabel('x (time in days)')
plt.ylabel('w(x) (flowers)')
plt.xticks(range(-10,15, 1))
plt.yticks(range(-200, 500, 20))
plt.grid()

# Plot the function
plt.plot(x,y, color='green')

# Plot the derivative
plt.plot(x,yd, color='purple')

plt.show()
```



Note that the green line represents the function, showing the number of flowers for 10 days before and after the fertilizer treatment. Before treatment, the number of flowers was in decline, and after treatment the flower bed started to recover.

The derivative function is shown in purple, and once again shows a linear change in slope. This time, the slope is increasing at a constant rate; and once again, the derivative function line crosses 0 at the lowest point in the function line (in other words, the slope changed from negative to positive when the flowers started to recover).

Critical Points

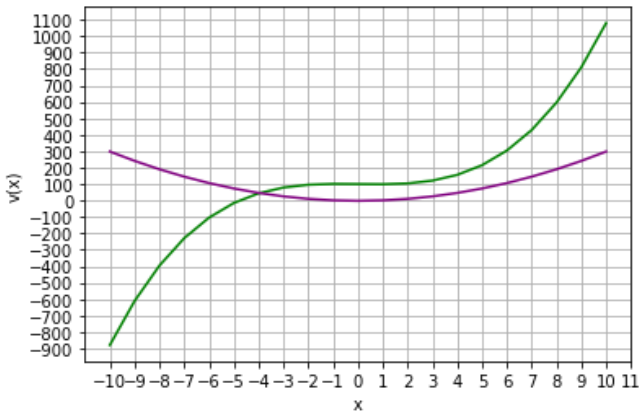
From what we've seen so far, it seems that there is a relationship between a function reaching an extreme value (a maximum or a minimum), and a derivative value of 0. This makes intuitive sense; the derivative represents the slope of the line, so when a function changes from a negative slope to a positive slope, or vice-versa, the derivative must pass through 0.

However, you need to be careful not to assume that just because the derivative is 0 at a given point, that this point represents the minimum or maximum of the function. For example, consider the following function:

$$v(x) = x^3 - 2x + 100$$

Run the following Python code to visualize this function and its corresponding derivative function:

In [5]:
```
%matplotlib inline

# Create function v
def v(x):
    return (x**3) - (2*x) + 100

def vd(x):
    return 3*(x**2) - 2

from matplotlib import pyplot as plt

# Create an array of x values to plot
x = list(range(-10, 11))

# Use the function to get the y values
y = [v(i) for i in x]

# Use the derivative function to get the derivative values
yd = [vd(i) for i in x]

# Set up the graph
plt.xlabel('x')
plt.ylabel('v(x)')
plt.xticks(range(-10,15, 1))
plt.yticks(range(-1000, 2000, 100))
plt.grid()

# Plot the function
plt.plot(x,y, color='green')

# Plot the derivative
plt.plot(x,yd, color='purple')

plt.show()
```



Note that in this case, the purple derivative function line passes through 0 as the green function line transitions from a *concave downwards* slope (a slope that is decreasing) to a *concave upwards* slope (a slope that is increasing). The slope flattens out to 0, forming a "saddle" before the it starts increasing.

What we can learn from this is that interesting things seem to happen to the function when the derivative is 0. We call points where the derivative crosses 0 *critical points*, because they indicate that the function is changing direction. When a function changes direction from positive to negative, it forms a peak (or a *local maximum*), when the function changes direction from negative to positive it forms a trough (or *local minimum*), and when it maintains the same overall direction but changes the concavity of the slope it creates an *inflexion point*.

Finding Minima and Maxima

A common use of calculus is to find minimum and maximum points in a function. For example, we might want to find out how many seconds it took for the kicked football to reach its maximum height, or how long it took for our fertilizer to be effective in reversing the decline of flower growth.

We've seen that when a function changes direction to create a maximum peak or a minimum trough, the derivative of the function is 0, so a step towards finding these extreme points might be to simply find all of the points in the function where the derivative is 0. For example, here's our function for the kicked football:

$$k(x) = -10x^2 + 100x + 3$$

From this, we've calculated the function for the derivative as:

$$k'(x) = -20x + 100$$

We can then solve the derivative equation for an f'(x) value of 0:

$$-20x + 100 = 0$$

We can remove the constant by subtracting 100 to both sides:

$$-20x = -100$$

Multiplying both sides by -1 gets rid of the negative values (this isn't strictly necessary, but makes the equation a little less confusing)

$$20x = 100$$

So:

$$x = 5$$

So we know that the derivative will be 0 when *x* is 5, but is this a minimum, a maximum, or neither? It could just be an inflexion point, or the entire function could be a constant value with a slope of 0) Without looking at the graph, it's difficult to tell.

Second Order Derivatives

The solution to our problem is to find the derivative of the derivative! Until now, we've found the derivative of a function, and indicated it as ***f'(x)***. Technically, this is known as the *prime* derivative; and it describes the slope of the function. Since the derivative function is itself a function, we can find its derivative, which we call the *second order* (or sometimes just *second*) derivative. This is indicated like this: ***f''(x)***.

So, here's our function for the kicked football:

$$k(x) = -10x^2 + 100x + 3$$

Here's the function for the prime derivative:

$$k'(x) = -20x + 100$$

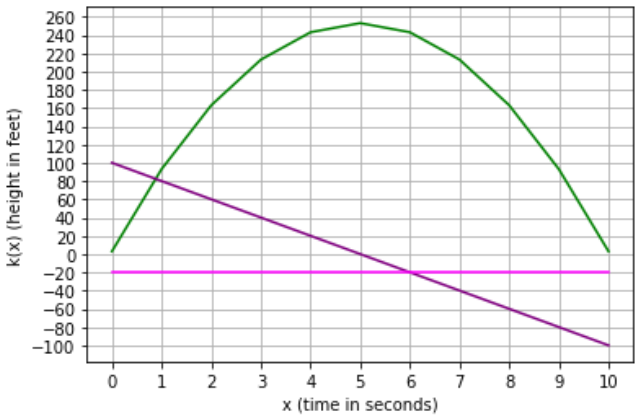
And using a combination of the power rule and the constant rule, here's the function for the second derivative:

$$k''(x) = -20$$

Now, without even drawing the graph, we can see that the second derivative has a constant value; so we know that the slope of the prime derivative is linear; and because it's a negative value, we know that it is decreasing. So when the prime derivative crosses 0, it we know that the slope of the function is decreasing linearly; so the point at *x=0* must be a maximum point.

Run the following code to plot the function, the prime derivative, and the second derivative for the kicked

In [6]:
```
%matplotlib inline

# Create function k
def k(x):
    return -10*(x**2) + (100*x)  + 3

def kd(x):
    return -20*x + 100

def k2d(x):
    return -20

from matplotlib import pyplot as plt

# Create an array of x values to plot
x = list(range(0, 11))

# Use the function to get the y values
y = [k(i) for i in x]

# Use the derivative function to get the k'(x) values
yd = [kd(i) for i in x]

# Use the 2-derivative function to get the k''(x)
y2d = [k2d(i) for i in x]

# Set up the graph
plt.xlabel('x (time in seconds)')
plt.ylabel('k(x) (height in feet)')
plt.xticks(range(0,15, 1))
plt.yticks(range(-200, 500, 20))
plt.grid()

# Plot the function
plt.plot(x,y, color='green')

# Plot  k'(x)
plt.plot(x,yd, color='purple')

# Plot k''(x)
plt.plot(x,y2d, color='magenta')

plt.show()
```



Let's take the same approach for the flower bed problem. Here's the function:

$$w(x) = x^2 + 2x + 7$$

Using the power rule and constant rule, gives us the prime derivative function:

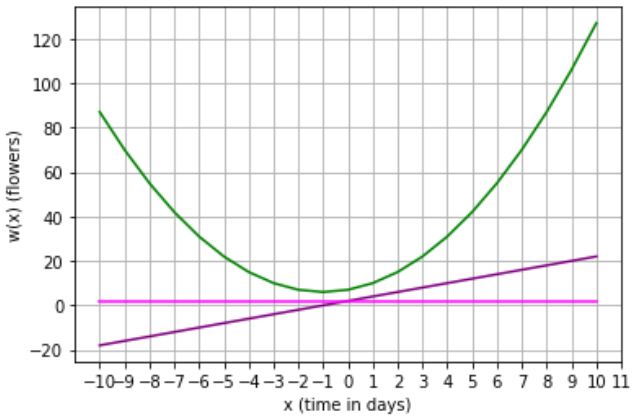
$$w'(x) = 2x + 2$$

Applying the power rule and constant rule to the prime derivative function gives us the second derivative function:

$$w''(x) = 2$$

Note that this time, the second derivative is a positive constant, so the prime derivative (which is the slope of the function) is increasing linearly. The point where the prime derivative crosses 0 must therefore be a minimum. Let's run the code below to check:

In [7]:
```
%matplotlib inline

# Create function w
def w(x):
    return (x**2) + (2*x) + 7

def wd(x):
    return 2*x + 2

def w2d(x):
    return 2

from matplotlib import pyplot as plt

# Create an array of x values to plot
x = list(range(-10, 11))

# Use the function to get the y values
y = [w(i) for i in x]

# Use the derivative function to get the w'(x) values
yd = [wd(i) for i in x]

# Use the 2-derivative function to get the w''(x) values
y2d = [w2d(i) for i in x]

# Set up the graph
plt.xlabel('x (time in days)')
plt.ylabel('w(x) (flowers)')
plt.xticks(range(-10,15, 1))
plt.yticks(range(-200, 500, 20))
plt.grid()

# Plot the function
plt.plot(x,y, color='green')

# Plot w'(x)
plt.plot(x,yd, color='purple')

# Plot w''(x)
plt.plot(x,y2d, color='magenta')

plt.show()
```



Critical Points that are *Not* Maxima or Minima

Of course, it's possible for a function to form a "saddle" where the prime derivative is zero at a point that is not a minimum or maximum. Here's an example of a function like this:

$$v(x) = x^3 - 6x^2 + 12x + 2$$

And here's its prime derivative:

$$v'(x) = 3x^2 - 12x + 12$$

Let's find a critical point where v'(x) = 0

$$3x^2 - 12x + 12 = 0$$

Factor the x-terms

$$3x(x - 4) = 12$$

Divide both sides by 3:

$$x(x - 4) = 4$$

Factor the x terms back again

$$x^2 - 4x = 4$$

Complete the square, step 1

$$x^2 - 4x + 4 = 0$$

Complete the square, step 2

$$(x - 2)^2 = 0$$

Find the square root:

$$x - 2 = \pm\sqrt{0}$$

$$x - 2 = +\sqrt{0} = 0, -\sqrt{0} = 0$$

v'(2) = 0 (only touches 0 once)

Is it a maximum or minimum? Let's find the second derivative:

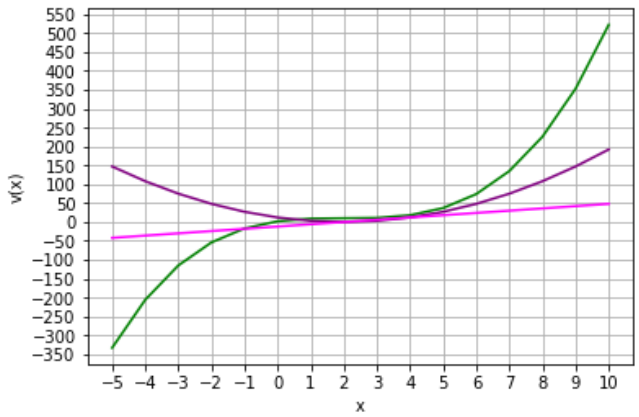
$$v''(x) = 6x - 12$$

So

$$v''(2) = 0$$

So it's neither negative or positive, so it's not a maximum or minimum.

In [8]:
```
%matplotlib inline

# Create function v
def v(x):
    return (x**3) - (6*(x**2)) + (12*x) + 2

def vd(x):
    return (3*(x**2)) - (12*x) + 12

def v2d(x):
    return (3*(2*x)) - 12

from matplotlib import pyplot as plt

# Create an array of x values to plot
x = list(range(-5, 11))

# Use the function to get the y values
y = [v(i) for i in x]

# Use the derivative function to get the derivative values
yd = [vd(i) for i in x]

# Use the derivative function to get the derivative values
y2d = [v2d(i) for i in x]

# Set up the graph
plt.xlabel('x')
plt.ylabel('v(x)')
plt.xticks(range(-10,15, 1))
plt.yticks(range(-2000, 2000, 50))
plt.grid()

# Plot the function
plt.plot(x,y, color='green')

# Plot the derivative
plt.plot(x,yd, color='purple')

# Plot the derivative
plt.plot(x,y2d, color='magenta')

plt.show()

print ("v(2) = " + str(v(2)))

print ("v'(2) = " + str(vd(2)))

print ("v''(2) = " + str(v2d(2)))
```




```
v(2) = 10
v'(2) = 0
```

Optimization

The ability to use derivatives to find minima and maxima of a function makes it a useful tool for scenarios where you need to optimize a function for a specific variable.

Defining Functions to be Optimized

For example, suppose you have decided to build an online video service that is based on a subscription model. You plan to charge a monthly subscription fee, and you want to make the most revenue possible. The problem is that customers are price-sensitive, so if you set the monthly fee too high, you'll deter some customers from signing up. Conversely, if you set the fee too low, you may get more customers, but at the cost of reduced revenue.

What you need is some kind of function that will tell you how many subscriptions you might expect to get based on a given fee. So you've done some research, and found a formula to indicate that the expected subscription volume (in thousands) can be calculated as 5-times the monthly fee subtracted from 100; or expressed as a function:

$$s(x) = -5x + 100$$

What you actually want to optimize is monthly revenue, which is simply the number of subscribers multiplied by the fee:

$$r(x) = s(x) \cdot x$$

We can combine ***s(x)*** into ***r(x)*** like this:

$$r(x) = -5x^2 + 100x$$

Finding the Prime Derivative

The function ***r(x)*** will return the expected monthly revenue (in thousands) for any proposed fee (*x*). What we need to do now is to find the fee that yields the maximum revenue. Fortunately, we can use a derivative to do that.

First, we need to determine the prime derivative of ***r(x)***, and we can do that easily using the power rule:

$$r'(x) = 2 \cdot -5x + 100$$

Which is:

$$r'(x) = -10x + 100$$

Find Critical Points

Now we need to find any critical points where the derivative is 0, as this could indicate a maximum:

$$-10x + 100 = 0$$

Let's isolate the *x* term:

$$-10x = -100$$

Both sides are negative, so we can mulitply both by -1 to make them positive without affecting the equation:

$$10x = 100$$

Now we can divide both sides by 10 to isolate *x*:

$$x = \frac{100}{10}$$

In [9]:
```
%matplotlib inline

# Create function s
def s(x):
    return (-5*x) + 100

# Create function r
def r(x):
    return s(x) * x

from matplotlib import pyplot as plt

# Create an array of x values to plot
x = list(range(0, 21))

# Use the function to get the y values
y = [r(i) for i in x]

# Set up the graph
plt.xlabel('x (monthly fee)')
plt.ylabel('r(x) (revenue in $,000)')
plt.xticks(range(0,22, 1))
plt.yticks(range(0, 600, 50))
plt.grid()

# Plot the function
plt.plot(x,y, color='green')

plt.show()
```