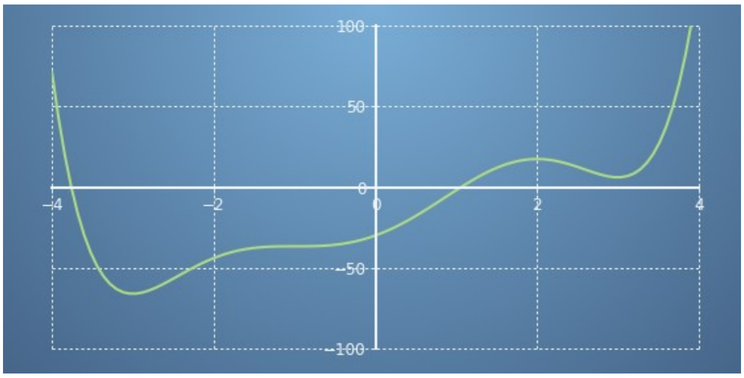
1. En este cuestionario, exploraremos el uso del método de Newton-Raphson para encontrar raíces.

1 / 1 punto

Considere la siguiente gráfica de una función,



Hay dos lugares por los que esta función pasa por cero, es decir, dos raíces, una está cerca $X = -4$ y el otro esta cerca $X = 1$.

Recuerde que si linealizamos sobre un punto particular X_0, we can ask what the value of the function is at the point $x_0 + \delta x$, a short distance away.

$$f(x_0 + \delta x) = f(x_0) + f'(x_0)\delta x$$

Then, if we assume that the function goes to zero somewhere nearby, we can re-arrange to find how far away, i.e. assume $f(x_0 + \delta x) = 0$ and solve for δx. This becomes,

$$\delta x = -\frac{f(x_0)}{f'(x_0)}$$

Since the function, $f(x)$ is not a line, this formula will (try) to get closer to the root, but won't exactly hit it. But this is OK, because we can repeat the process from the new starting point to get even closer,

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)},$$

This is the Newton-Raphson method, and it (or a variant) is used widely to find the roots of functions.

For the graph we showed above, the equation of the function is,

$$f(x) = \frac{x^6}{6} - 3x^4 - \frac{2x^3}{3} + \frac{27x^2}{2} + 18x - 30.$$

We'll explore the Newton-Raphson method for this function in this quiz, when it works, and how it can go wrong.

To start, differentiate the function $f(x)$, as we'll need $f'(x)$ later on.

(Type your answer as you would Python code, i.e with * to multiply and ** to raise to a power. e.g., 4*x**3 - 2*x**2/5)

$x^5 - 12x^3 - 2x^2 + 27x + 18$

(x**5)-(12*(x**3))- (2*x**2)+(27*x)+18

Correcto
Exactly, this is power rule differentiation.

2. We'll first try to find the location of the root near $x = 1$.

1 / 1 punto

By using $x_0 = 1$ as a starting point and calculating $-f(1)/f'(1)$ by hand, find the first iteration of the Newton-Raphson method, i.e., find x_1.

Give your answer to 3 decimal places.

1.062

Correcto

3. Let's use code to find the other root, near $x = -4$.

1 / 1 punto

Complete the d_f function in the code block with your answer to Q1, i.e. with $f'(x)$. The code block will then perform iterations of the Newton-Raphson method.

```
def f (x) :
  return x**6/6 - 3*x**4 - 2*x**3/3 + 27*x**2/2 + 18*x - 30

def d_f (x) :
  return (x**5)-(12*(x**3))- (2*x**2)+(27*x)+18

x = 1.99

d = {"x" : [x], "f(x)": [f(x)]}
for i in range(0, 20):
  x = x - f(x) / d_f(x)
  d["x"].append(x)
  d["f(x)"].append(f(x))

pd.DataFrame(d, columns=['x', 'f(x)'])
```

Ejecutar

Restablecer

```
            x          f(x)
0    1.990000  1.733108e+01
1  -36.474613  3.871975e+08
2  -30.422744  1.296022e+08
3  -25.384916  4.337012e+07
4  -21.193182  1.450848e+07
5  -17.707798  4.851113e+06
6  -14.812568  1.620886e+06
7  -12.410978  5.410173e+05
8  -10.423021  1.802993e+05
9   -8.782600  5.994279e+04
10  -7.435499  1.985174e+04
11  -6.337977  6.530275e+03
12  -5.456155  2.120398e+03
13  -4.766625  6.692936e+02
14  -4.258508  1.970466e+02
15  -3.933902  4.793270e+01
16  -3.788984  6.712292e+00
17  -3.761154  2.123257e-01
18  -3.760215  2.351946e-04
19  -3.760214  2.895888e-10
20  -3.760214  4.263256e-14
```

What is the x value of the root near $x = -4$? (to 3 decimal places.)

-3.760

Correcto
Observe that the function converges in just a few iterations.

4. Let's explore where things can go wrong with Newton-Raphson.

1 / 1 punto

Since the step size is given by $\delta x = -f(x)/f'(x)$, this can get big when f'(x) is very small. In fact $f'(x)$ is exactly zero at turning points of $f(x)$. This is where Newton-Raphson behaves the worst since the step size is infinite.

Use the code block in the previous question for a starting point of $x_0 = 1.99$ and observe what happens.

Select all true statements.

- [x] The method takes over 15 iterations to converge.

> **Correcto**
> Contrast this to a starting value of -4 or 1 where convergence was very quick.

- [ ] The method diverges to infinity.
- [ ] None of the other statements are true.
- [ ] The method converges to the root nearest $x = 1$
- [ ] The method does not converge, instead oscillates without settling.
- [x] The method converges to the root nearest $x = -4$

> **Correcto**
> Note that this is not the nearest root to the starting point.

5. Some starting points on the curve do not converge, nor do they diverge, but oscillate without settling. Try $x_0 = 3.1$ as a starting point; it does just this.

1 / 1 punto

Again, this is behaviour that happens in areas where the curve is not well described by a straight line - therefore our initial linearisation assumption was not a good one for such a starting point.

Use the code block from previously to observe this.

En la práctica, a menudo no necesitará crear manualmente métodos de optimización, ya que se pueden llamar desde bibliotecas, como scipy. Use el bloque de código a continuación para probar $X_0 = 3.1$.

```
de  scipy  importar  optimizar

def  f (x) :
  volver  x** 6 / 6  -  3 *x** 4  -  2 *x** 3 / 3  +  27 *x** 2 / 2  +  18 *x -

x0 =  3,1
optimizar.newton(f, x0)
```

Ejecutar

Restablecer

```
1.063070629709697
```

¿Se asentó en una raíz?

- ( ) No, el método devolvió un error.
- ( ) Sí, a la raíz más cercana $X = -4$.
- ( ) No, el método divergió.
- (•) Sí, a la raíz más cercana $X = 1$.

Correcto

Eventualmente, este complicado punto de partida se resuelve.