InstitutoTecnológico y de Estudios Superiores de Occidente

Maestría Ciencia de Datos

Investigación, Desarrollo e Innovación II

Tarea 1: Método de Raphson para una variable

Estudiante: Daniel Nuño
Titular: Fernando Becerra
10 Septiembre 2021

Realice código en Python que, recibiendo una función f dada, un valor inicial x0 y una exactitud (error) dado E , encuentre una aproximación de exactitud menor a E para x cuando f(x)=0 usando el método de Newton-Raphson. Asegúrese que cuenta el número de iteraciones realizadas.

Use su código para resolver los siguientes ejercicios (en todos los casos indique el(los) valor(es) inicial(es) que utilizó y el número de iteraciones que fueron necesarias para alcanzar la respuesta).

Escriba sus respuestas con 10 cifras significativas.

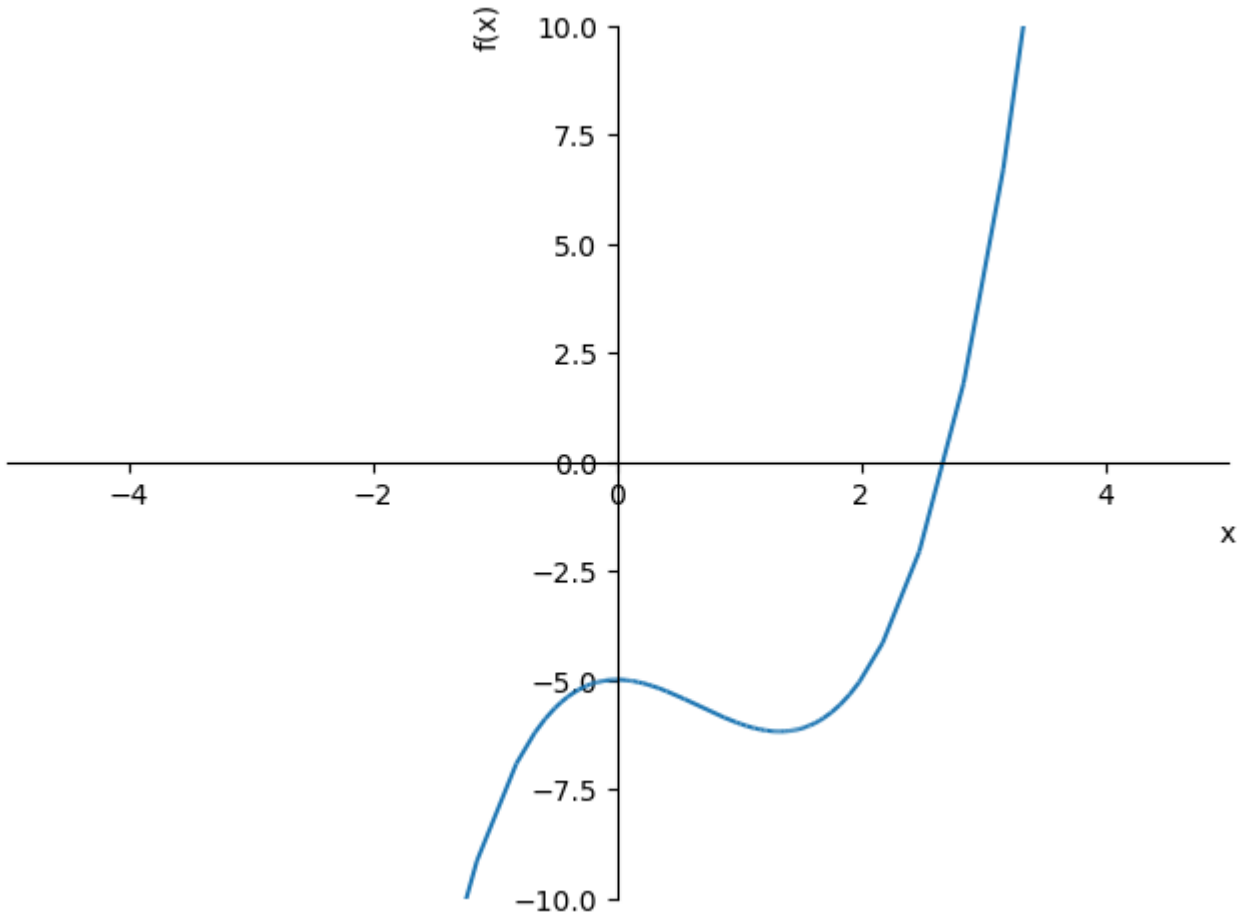
Aplique el método de Newton-Raphson para encontrar todas las soluciones exactas dentro de 10e−4.

Funcón 1:

$$f(x) = x^3 - 2x^2 - 5$$

In [1]:

```
import sympy as sp

x = sp.symbols('x')
sp.init_printing(use_unicode=True)
f = x**3 - 2*x**2 - 5
f_derivada = f.diff(x,1)
x0 = 5
exactitud = 10e-4
iteraciones = 0
list_resultados = [x0]
error = 1
sp.plotting.plot(f,  xlim=(-5, 5), ylim=(-10,10))
f
```




```
C:\Users\nuno\AppData\Local\Continuum\anaconda3\lib\site-packages\IPython\lib\latextool
s.py:126: MatplotlibDeprecationWarning:
The to_png function was deprecated in Matplotlib 3.4 and will be removed two minor relea
ses later. Use mathtext.math_to_image instead.
  mt.to_png(f, s, fontsize=12, dpi=dpi, color=color)
C:\Users\nuno\AppData\Local\Continuum\anaconda3\lib\site-packages\IPython\lib\latextool
s.py:126: MatplotlibDeprecationWarning:
The to_rgba function was deprecated in Matplotlib 3.4 and will be removed two minor rele
ases later. Use mathtext.math_to_image instead.
  mt.to_png(f, s, fontsize=12, dpi=dpi, color=color)
C:\Users\nuno\AppData\Local\Continuum\anaconda3\lib\site-packages\IPython\lib\latextool
s.py:126: MatplotlibDeprecationWarning:
The to_mask function was deprecated in Matplotlib 3.4 and will be removed two minor rele
ases later. Use mathtext.math_to_image instead.
  mt.to_png(f, s, fontsize=12, dpi=dpi, color=color)
C:\Users\nuno\AppData\Local\Continuum\anaconda3\lib\site-packages\IPython\lib\latextool
s.py:126: MatplotlibDeprecationWarning:
The MathtextBackendBitmap class was deprecated in Matplotlib 3.4 and will be removed two
minor releases later. Use mathtext.math_to_image instead.
  mt.to_png(f, s, fontsize=12, dpi=dpi, color=color)
```

Out[1]: $x^3 - 2x^2 - 5$

Observamos que tiene una raíz entre 0 y 2.5. Comenzamos X0 en 3.

In [2]:
```
while error >= exactitud:
    iteraciones += 1
    y = f.evalf(subs={x: x0})
    y_derivada = f_derivada.evalf(subs={x: x0})
    x1 = sp.N(x0 - y/y_derivada, 10)
```

```
    list_resultados.append(x1)
    error = abs(x1 - x0)
    x0 = x1

print('El número de iteraciones fue: ' + str(iteraciones))
print('Usando x inicial: ' + str(list_resultados[0]))
print('La raíz es cercana a: ' + str(list_resultados[-1]))
```

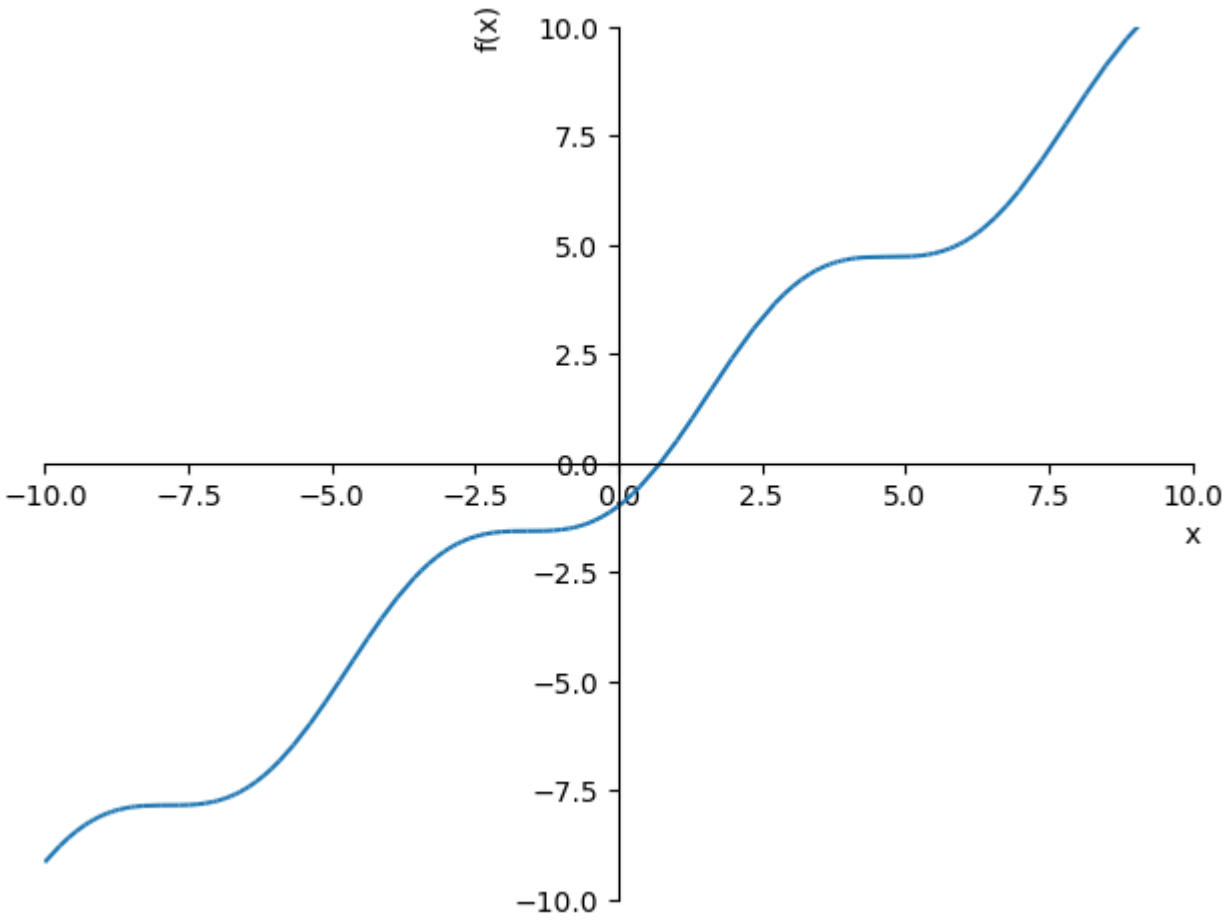
```
El número de iteraciones fue: 6
Usando x inicial: 5
La raíz es cercana a: 2.690647448
```

Funcón 2:

$$f(x) = x - cos(x)$$

In [3]:

```
x = sp.symbols('x')
f = x - sp.cos(x)
f_derivada = f.diff(x,1)
exactitud = 10e-4
iteraciones = 0
error = 1
sp.plotting.plot(f,  xlim=(-10, 10), ylim=(-10,10))
f
```



Out[3]: $x - \cos(x)$

Observamos que tiene una raíz entre 0 y 2. Comenzamos X0 en 1.

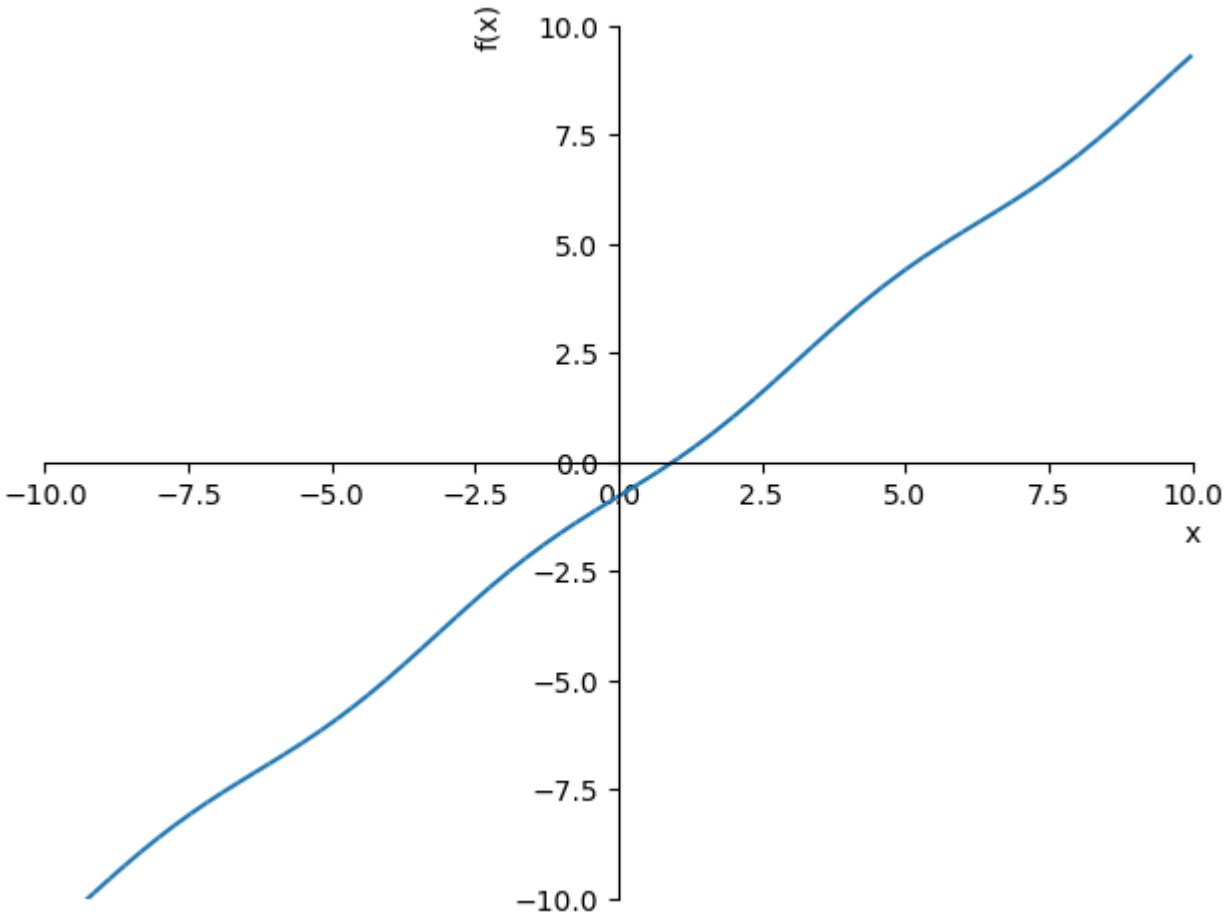
In [4]:
```
x0 = 1
list_resultados = [x0]

while error >= exactitud:
    iteraciones += 1
    y = f.evalf(subs={x: x0})
    y_derivada = f_derivada.evalf(subs={x: x0})
    x1 = sp.N(x0 - y/y_derivada, 10)
    list_resultados.append(x1)
    error = abs(x1 - x0)
    x0 = x1

print('El número de iteraciones fue: ' + str(iteraciones))
print('Usando x inicial: ' + str(list_resultados[0]))
print('La raíz es cercana a: ' + str(list_resultados[-1]))
```

```
El número de iteraciones fue: 3
Usando x inicial: 1
La raíz es cercana a: 0.7390851334
```

Funcón 3:

$$f(x) = x - 0.8 - 0.2sp.sin(x)$$

In [5]:
```
x = sp.symbols('x')
f = x - 0.8 - 0.2*sp.sin(x)
f_derivada = f.diff(x,1)
exactitud = 10e-4
iteraciones = 0
error = 1
sp.plotting.plot(f,  xlim=(-10, 10), ylim=(-10,10))
f
```



Out[5]: $x - 0.2\sin(x) - 0.8$

Observamos que tiene una raíz entre 0 y 2. Comenzamos X0 en 1.

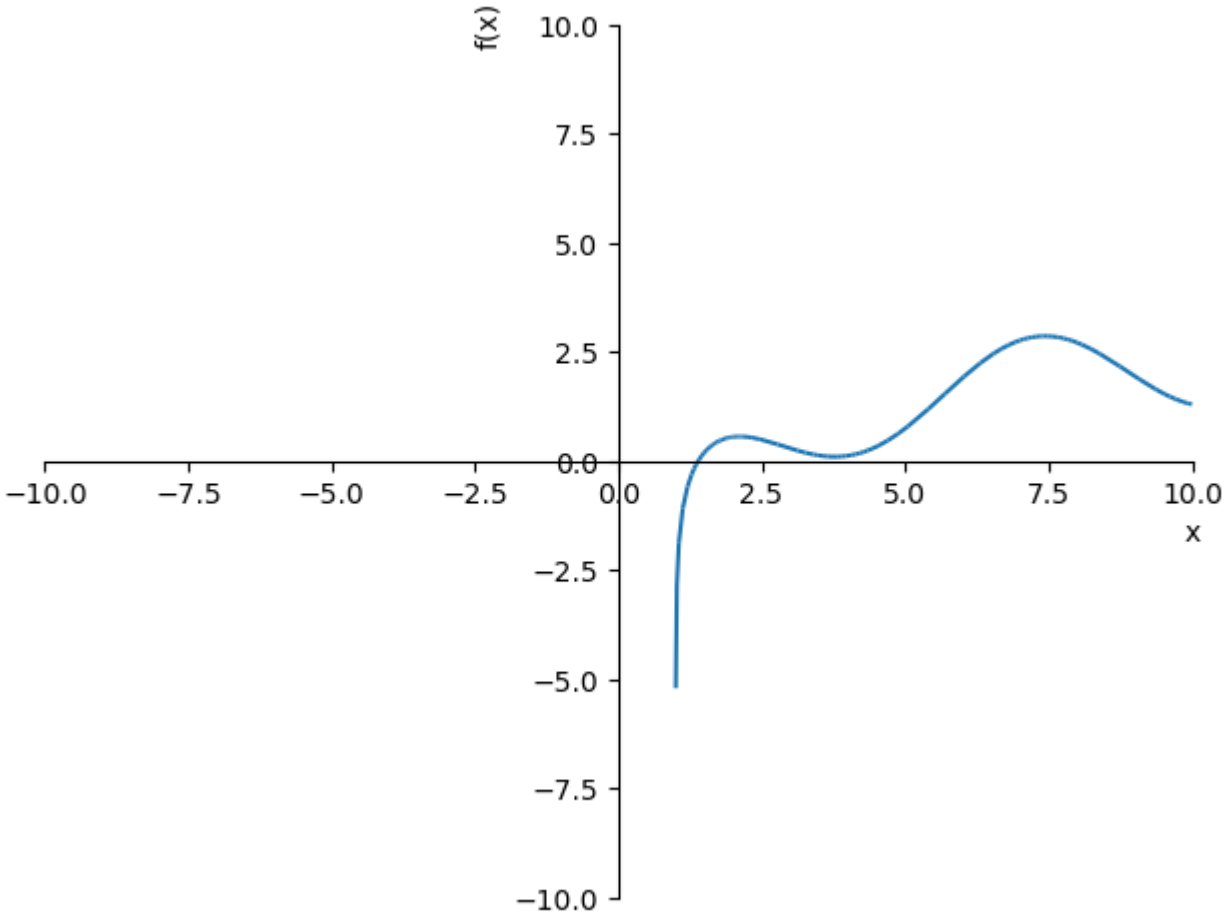
In [6]:
```
x0 = 1
list_resultados = [x0]

while error >= exactitud:
    iteraciones += 1
    y = f.evalf(subs={x: x0})
    y_derivada = f_derivada.evalf(subs={x: x0})
    x1 = sp.N(x0 - y/y_derivada, 10)
    list_resultados.append(x1)
    error = abs(x1 - x0)
    x0 = x1

print('El número de iteraciones fue: ' + str(iteraciones))
print('Usando x inicial: ' + str(list_resultados[0]))
print('La raíz es cercana a: ' + str(list_resultados[-1]))
```

```
El número de iteraciones fue: 2
Usando x inicial: 1
La raíz es cercana a: 0.9643338890
```

Funcón 4:

$$f(x) = ln(x-1) + cos(x-1)$$

In [7]:
```
x = sp.symbols('x')
f = sp.log(x-1) + sp.cos(x-1)
f_derivada = f.diff(x,1)
exactitud = 10e-4
iteraciones = 0
error = 1
sp.plotting.plot(f,  xlim=(-10, 10), ylim=(-10,10))
f
```



Out[7]: $\log(x-1) + \cos(x-1)$

Observamos que tiene una raíz entre 0 y 2. Comenzamos X0 en 1.3.

In [8]:
```
x0 = 1.3
list_resultados = [x0]

while error >= exactitud:
    iteraciones += 1
    y = f.evalf(subs={x: x0})
    y_derivada = f_derivada.evalf(subs={x: x0})
    x1 = sp.N(x0 - y/y_derivada, 10)
    list_resultados.append(x1)
    error = abs(x1 - x0)
    x0 = x1

print('El número de iteraciones fue: ' + str(iteraciones))
print('Usando x inicial: ' + str(list_resultados[0]))
print('La raíz es cercana a: ' + str(list_resultados[-1]))
```

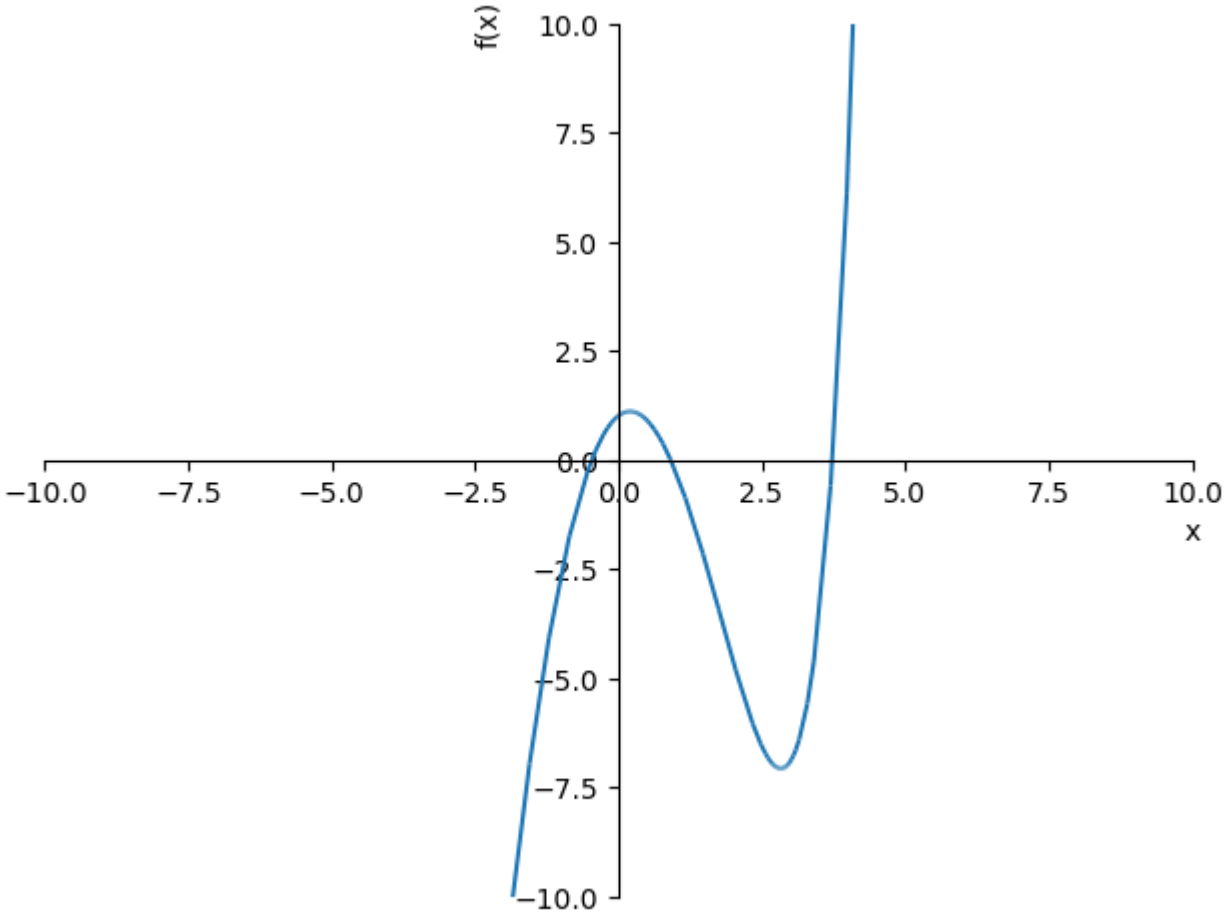
```
El número de iteraciones fue: 3
Usando x inicial: 1.3
La raíz es cercana a: 1.397748164
```

Funcón 5:

$$f(x) = e^x - 3x^2$$

In [9]:

```
x = sp.symbols('x')
f = sp.exp(x) - 3*x**2
f_derivada = f.diff(x,1)
exactitud = 10e-4
iteraciones = 0
error = 1
sp.plotting.plot(f,  xlim=(-10, 10), ylim=(-10,10))
f
```



Out[9]: $-3x^2 + e^x$

Observamos que tiene una raíz 3 raices entre -1 y 5. Comenzamos X0 en 0 para buscar la primer raíz.

In [10]:

```
x0 = 0
list_resultados = [x0]

while error >= exactitud:
```

```
    iteraciones += 1
    y = f.evalf(subs={x: x0})
    y_derivada = f_derivada.evalf(subs={x: x0})
    x1 = sp.N(x0 - y/y_derivada, 10)
    list_resultados.append(x1)
    error = abs(x1 - x0)
    x0 = x1

print('El número de iteraciones fue: ' + str(iteraciones))
print('Usando x inicial: ' + str(list_resultados[0]))
print('La raíz es cercana a: ' + str(list_resultados[-1]))
```

```
El número de iteraciones fue: 5
Usando x inicial: 0
La raíz es cercana a: -0.4589622742
```

Ahora buscamos la segunda raíz y comenzamos X0 en 1.5.

In [11]:

```
x0 = 1.5
list_resultados = [x0]

while error >= exactitud:
    iteraciones += 1
    y = f.evalf(subs={x: x0})
    y_derivada = f_derivada.evalf(subs={x: x0})
    x1 = sp.N(x0 - y/y_derivada, 10)
    list_resultados.append(x1)
    error = abs(x1 - x0)
    x0 = x1

print('El número de iteraciones fue: ' + str(iteraciones))
print('Usando x inicial: ' + str(list_resultados[0]))
print('La raíz es cercana a: ' + str(list_resultados[-1]))
```

```
El número de iteraciones fue: 5
Usando x inicial: 1.5
La raíz es cercana a: 1.5
```

Ahora buscamos la tercera raíz y comenzamos X0 en 4.

In [12]:

```
x0 = 4
list_resultados = [x0]

while error >= exactitud:
    iteraciones += 1
    y = f.evalf(subs={x: x0})
    y_derivada = f_derivada.evalf(subs={x: x0})
    x1 = sp.N(x0 - y/y_derivada, 10)
    list_resultados.append(x1)
    error = abs(x1 - x0)
    x0 = x1

print('El número de iteraciones fue: ' + str(iteraciones))
print('Usando x inicial: ' + str(list_resultados[0]))
print('La raíz es cercana a: ' + str(list_resultados[-1]))
```

```
El número de iteraciones fue: 5
Usando x inicial: 4
La raíz es cercana a: 4
```

Encontrar la raiz de:

$$\sqrt{(5)}$$

In [13]:
```
x = sp.symbols('x')
f = sp.sqrt(5)
exactitud = 10e-4
iteraciones = 0
error = 1
```

In [14]:
```
number = 5
x0 = number
list_resultados = [sp.N(0.5 * (x0 + number/x0), 10)]

while error >= exactitud:
    iteraciones += 1
    x1 = sp.N(0.5 * (x0 + number/x0), 10)
    list_resultados.append(x1)
    error = abs(x1 - x0)
    x0 = x1

print('El número de iteraciones fue: ' + str(iteraciones))
print('Usando x inicial: ' + str(list_resultados[0]))
print('La raíz es cercana a: ' + str(list_resultados[-1]))
```

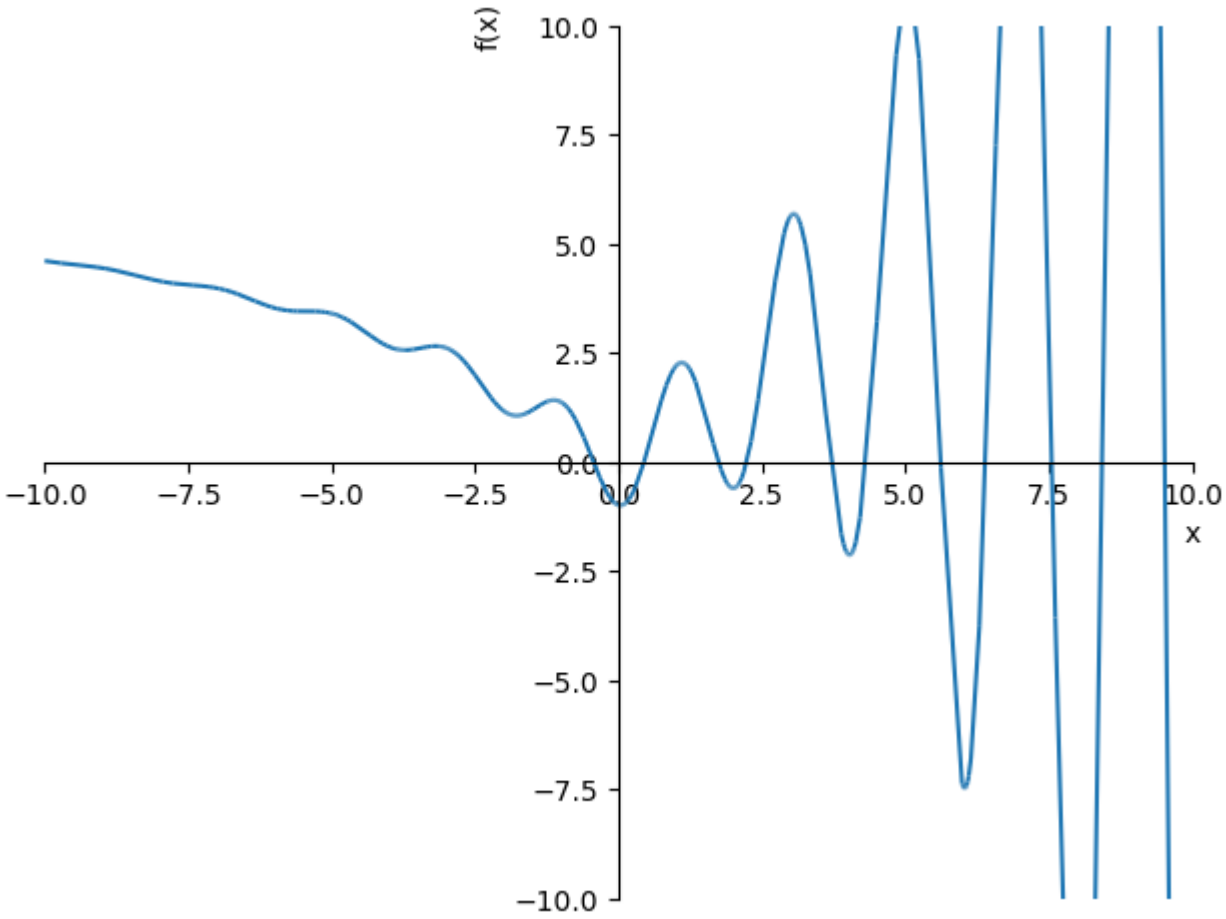
```
El número de iteraciones fue: 5
Usando x inicial: 3.000000000
La raíz es cercana a: 2.236067978
```

Encuentre el único cero negativo de

$$f(x) = ln(x^2 + 1) - e^(0.4x)cos(\pi x)$$

con exactitud de 10e−6 usando Newton-Raphson

In [15]:
```
x = sp.symbols('x')
f = sp.log(x**2+1) - sp.exp(0.4*x) * sp.cos(sp.pi*x)
f_derivada = f.diff(x,1)
exactitud = 10e-6
iteraciones = 0
error = 1
sp.plotting.plot(f,  xlim=(-10, 10), ylim=(-10,10))
f
```



Out[15]: $-e^{0.4x}\cos(\pi x) + \log\left(x^2+1\right)$

In [16]:
```
x0 = -2
list_resultados = [x0]

while error >= exactitud:
    iteraciones += 1
    y = f.evalf(subs={x: x0})
    y_derivada = f_derivada.evalf(subs={x: x0})
    x1 = sp.N(x0 - y/y_derivada, 10)
    list_resultados.append(x1)
    error = abs(x1 - x0)
    x0 = x1

print('El número de iteraciones fue: ' + str(iteraciones))
print('Usando x inicial: ' + str(list_resultados[0]))
print('La raíz es cercana a: ' + str(list_resultados[-1]))
```

```
El número de iteraciones fue: 6
Usando x inicial: -2
La raíz es cercana a: -0.4341430473
```

código usado:

In [17]:
```
%%script echo skipping

x = sp.symbols('x')
f = sp.log(x**2+1) - sp.exp(0.4*x) * sp.cos(sp.pi*x)
```

```
f_derivada = f.diff(x,1)
exactitud = 10e-6
iteraciones = 0
error = 1

x0 = -2
list_resultados = [x0]

while error >= exactitud:
    iteraciones += 1
    y = f.evalf(subs={x: x0})
    y_derivada = f_derivada.evalf(subs={x: x0})
    x1 = sp.N(x0 - y/y_derivada, 10)
    list_resultados.append(x1)
    error = abs(x1 - x0)
    x0 = x1

print('El número de iteraciones fue: ' + str(iteraciones))
print('Usando x inicial: ' + str(list_resultados[0]))
print('La raíz es cercana a: ' + str(list_resultados[-1]))
```

```
Couldn't find program: 'echo'
```

In [ ]: