Ejercicios para practicar.

1. *Punto fijo*

   Determinar una solución para $f(x) = 2sen\sqrt{x} - x$ sujeto a $x_o = 0.5$.

```
In [6]: FixedP(0.5, 0.1, 100)

          f(x) = -x + 2*sin(sqrt(x))
          x = 2*sin(sqrt(x))

1. P = 1.29927387816012 Er = 61.5169666377008
2. P = 1.81714750413750 Er = 28.4992618815046
3. P = 1.95057391728976 Er = 6.84036692839891
4. P = 1.96974251333848 Er = 0.973152374938294
5. P = 1.97206888127395 Er = 0.117965855937141
6. P = 1.97234417827267 Er = 0.0139578579516519

Out[6]: 1.97234417827267
```

2. *Interpolación*

   Considera los siguientes datos: $x = 8, x = 9, x = 11$. Estima $\log_{10} 10$.

   a. *Usando Lagrange de orden 2.*

```
In [5]: f, e = Lagrange(xInput, yInput), 10
        print("\ng(", e, ") ≈ ", N(f.subs(x, e)), sep = "")

3 points:
        f(8) = 0.903089986991944
        f(9) = 0.954242509439325
        f(11) = 1.04139268515823

Polynomial
0.903089986991944*(x - 9)/(8 - 9)*(x - 11)/(8 - 11) + 0.954242509439325*(x - 8)/(9 - 8)*(x - 11)/
(9 - 11) + 1.04139268515823*(x - 8)/(11 - 8)*(x - 9)/(11 - 9)

Simplified
-0.00252581152930949*x**2 + 0.0940913184456432*x + 0.312011377302608

By Powers
-0.00252581152930953*x**2 + 0.0940913184456438*x + 0.312011377302607

g(10) ≈ 1.00034340882809
```

b. *Usando Newton de orden 3. ($x = 12$).*

In [9]:
```
f, e = Newton(xInput, yInput), 10
print("\nf(", e, ") ≈ ", N(f.subs(x, e)), sep = "")
```

```
4 points:
        f(8) = 0.903089986991944
        f(9) = 0.954242509439325
        f(11) = 1.04139268515823
        f(12) = 1.07918124604762

Polynomial
0.903089986991944 + 0.0511525224473813*(x-8) + -0.00252581152931040*(x-8)*(x-9) + 0.0001492423014
90068*(x-8)*(x-9)*(x-11)

Simplified
0.000149242301490068*x**3 - 0.0067045959710323*x**2 + 0.132745074531586*x + 0.193811474522411

By Powers
0.000149242301490068*x**3 - 0.0067045959710323*x**2 + 0.132745074531586*x + 0.193811474522411

f(10) ≈ 1.00004492422511
```

c. *Utilizando el método de Hermite*

In [25]:
```
f, e = Hermite(xInput, yInput, dInput), 10
print("\nf(", e, ") ≈ ", N(f.subs(x, e)), sep = "")
```

```
3 points:
        f(8) = 0.903089986991944        f'(8) = 0.0542868102379065
        f(9) = 0.954242509439325        f'(9) = 0.0482549424336946
        f(11) = 1.04139268515823        f'(11) = 0.0394813165366593

Polynomial
0.903089986991944 + 0.0542868102379065*(x-8) + -0.00313428779052514*(x-8)*(x-8) + 0.0002367077768
38457*(x-8)*(x-8)*(x-9) + -1.69411782167711e-5*(x-8)*(x-8)*(x-9)*(x-9) + 1.27334503617518e-6*(x-
8)*(x-8)*(x-9)*(x-9)*(x-11)

Simplified
1.27334503617518e-6*x**5 - 7.42417048446542e-5*x**4 + 0.00184029728040204*x**3 - 0.02556960343520
77*x**2 + 0.236032292545021*x - 0.0285768881824631

By Powers
1.27334503617518e-6*x**5 - 7.42417048446542e-5*x**4 + 0.00184029728040204*x**3 - 0.02556960343520
77*x**2 + 0.236032292545021*x - 0.0285768881824631

f(10) ≈ 1.00000042932000
```

3. *Sistemas*

Resuelve por Jacobi o *Gauss-Seidel.*

$$\begin{pmatrix} 10x_1 & 2x_2 & -x_3 \\ -3x_1 & -6x_2 & 2x_3 \\ x_1 & x_2 & 5x_3 \end{pmatrix} = \begin{pmatrix} 27 \\ -61.5 \\ -21.5 \end{pmatrix}$$

```python
x = GaussSeidel(Matrix, Independent, 2, 0.1)
print(x)
```

```
3x3 System:

        [10 2 -1 = 27]
        [-3 -6 2 = -61.5]
        [1 1 5 = -21.5]

x[0] = [2.7, 8.9, -6.62]
x[1] = [0.2579999999999999, 7.914333333333333, -5.934466666666666]
[0.5236866666666666, 8.010001111111112, -6.0067375555555556]
```