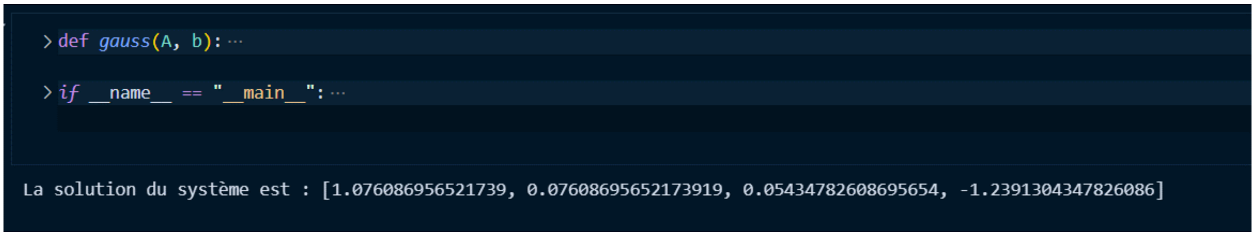
AMAVIGAN Ayi Mawuli Hénoc GL S3

EXAMEN

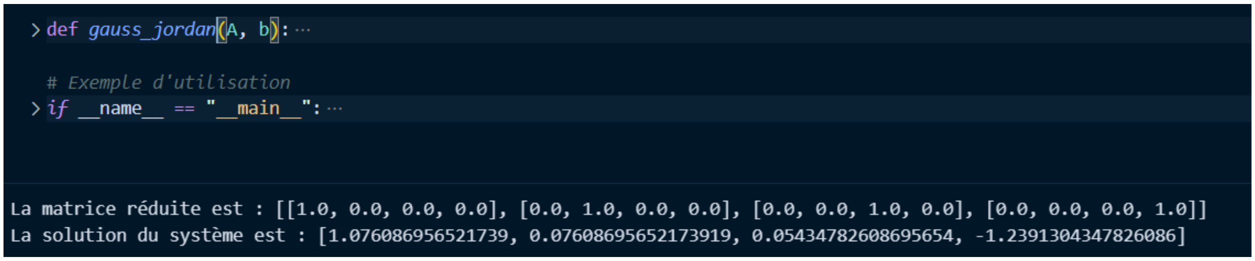
1



```
> def gauss(A, b): ···

> if __name__ == "__main__": ···
```

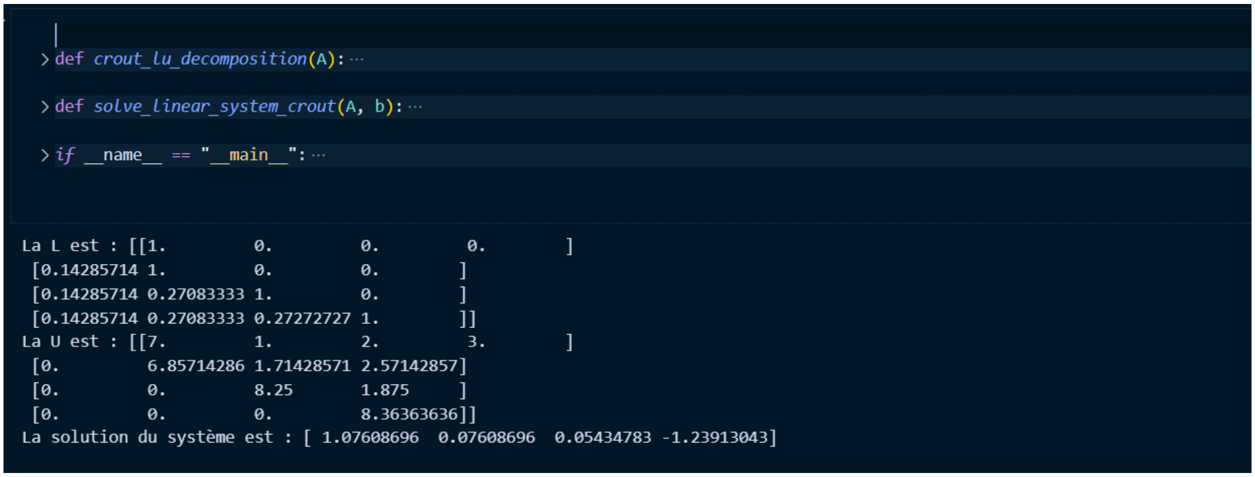
```
La solution du système est : [1.076086956521739, 0.07608695652173919, 0.05434782608695654, -1.2391304347826086]
```



```
> def gauss_jordan(A, b): ···

  # Exemple d'utilisation
> if __name__ == "__main__": ···
```

```
La matrice réduite est : [[1.0, 0.0, 0.0, 0.0], [0.0, 1.0, 0.0, 0.0], [0.0, 0.0, 1.0, 0.0], [0.0, 0.0, 0.0, 1.0]]
La solution du système est : [1.076086956521739, 0.07608695652173919, 0.05434782608695654, -1.2391304347826086]
```

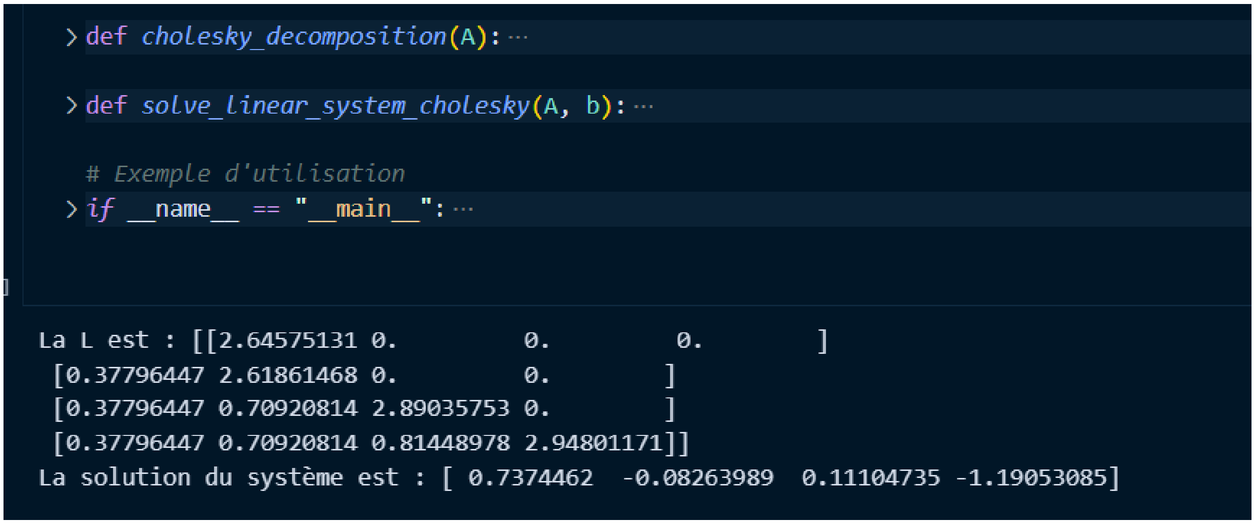


```
> def crout_lu_decomposition(A): ···

> def solve_linear_system_crout(A, b): ···

> if __name__ == "__main__": ···
```

```
La L est : [[1.         0.         0.         0.        ]
 [0.14285714 1.         0.         0.        ]
 [0.14285714 0.27083333 1.         0.        ]
 [0.14285714 0.27083333 0.27272727 1.        ]]
La U est : [[7.         1.         2.         3.        ]
 [0.         6.85714286 1.71428571 2.57142857]
 [0.         0.         8.25       1.875     ]
 [0.         0.         0.         8.36363636]]
La solution du système est : [ 1.07608696  0.07608696  0.05434783 -1.23913043]
```

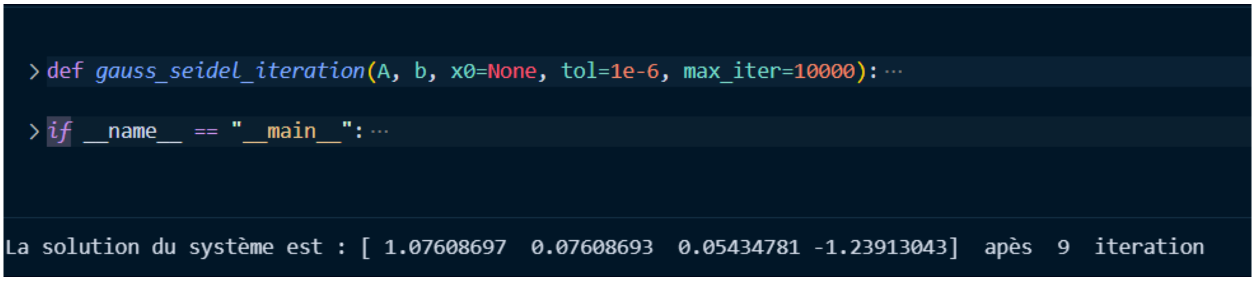


```
def cholesky_decomposition(A): ···

def solve_linear_system_cholesky(A, b): ···

# Exemple d'utilisation
if __name__ == "__main__": ···
```

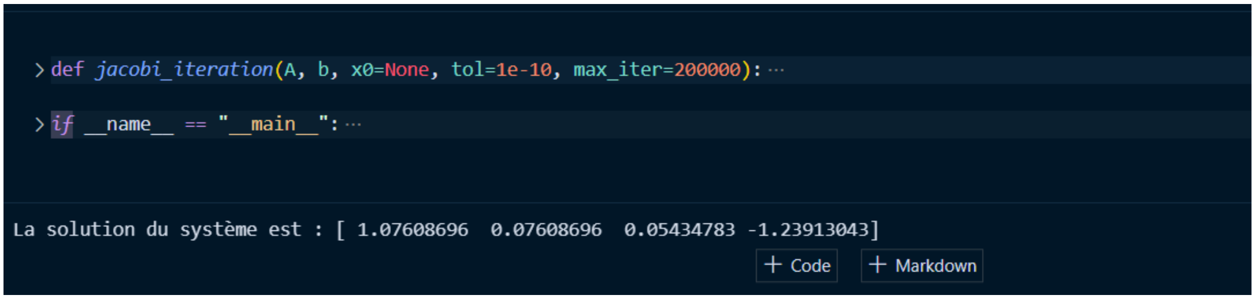
```
La L est : [[2.64575131 0.         0.         0.        ]
 [0.37796447 2.61861468 0.         0.        ]
 [0.37796447 0.70920814 2.89035753 0.        ]
 [0.37796447 0.70920814 0.81448978 2.94801171]]
La solution du système est : [ 0.7374462  -0.08263989  0.11104735 -1.19053085]
```



```
def gauss_seidel_iteration(A, b, x0=None, tol=1e-6, max_iter=10000): ···

if __name__ == "__main__": ···
```

```
La solution du système est : [ 1.07608697  0.07608693  0.05434781 -1.23913043]  apès  9  iteration
```

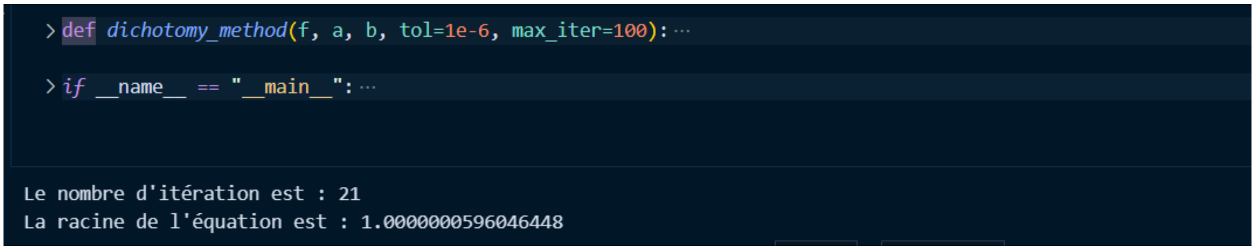


```
def jacobi_iteration(A, b, x0=None, tol=1e-10, max_iter=200000): ···

if __name__ == "__main__": ···
```

```
La solution du système est : [ 1.07608696  0.07608696  0.05434783 -1.23913043]
```

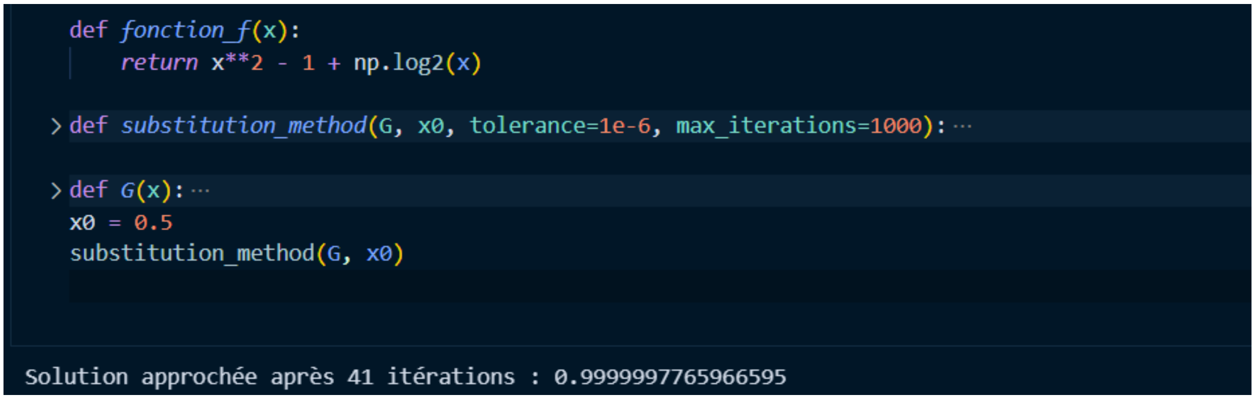
2



```
def dichotomy_method(f, a, b, tol=1e-6, max_iter=100): ···

if __name__ == "__main__": ···
```

```
Le nombre d'itération est : 21
La racine de l'équation est : 1.0000000596046448
```

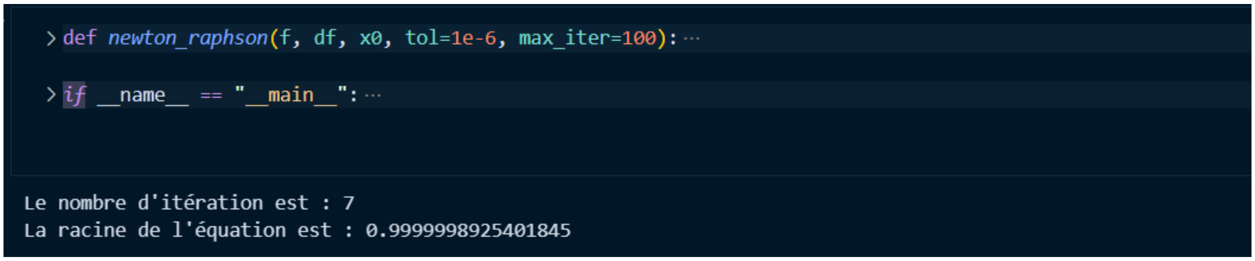


```
def fonction_f(x):
    return x**2 - 1 + np.log2(x)

def substitution_method(G, x0, tolerance=1e-6, max_iterations=1000): ···

def G(x): ···
x0 = 0.5
substitution_method(G, x0)
```

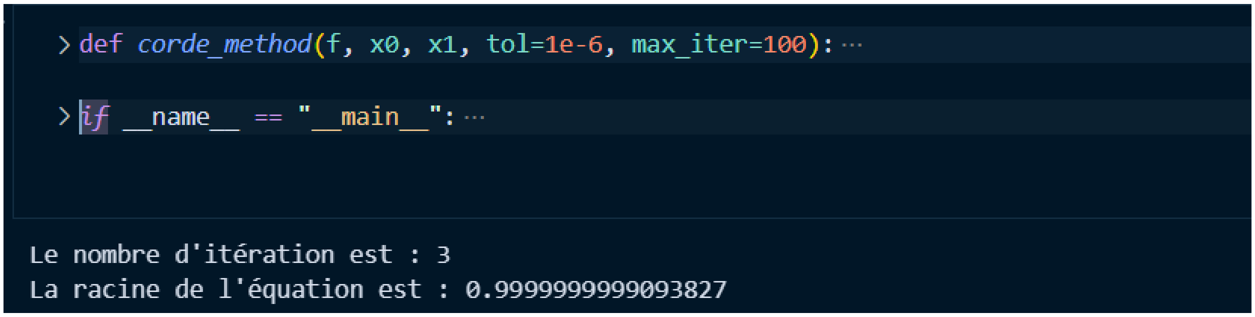
```
Solution approchée après 41 itérations : 0.9999997765966595
```



```
def newton_raphson(f, df, x0, tol=1e-6, max_iter=100): ···

if __name__ == "__main__": ···
```

```
Le nombre d'itération est : 7
La racine de l'équation est : 0.9999998925401845
```

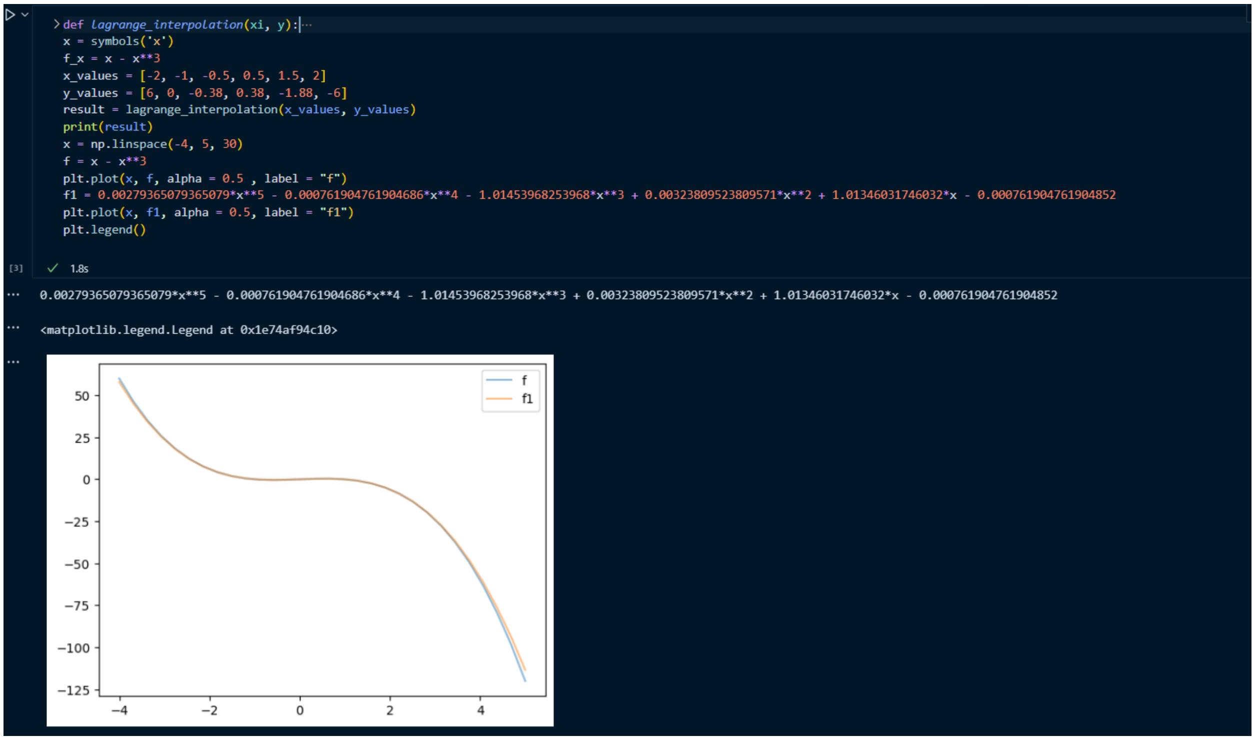


```
def corde_method(f, x0, x1, tol=1e-6, max_iter=100): ···

if __name__ == "__main__": ···
```

```
Le nombre d'itération est : 3
La racine de l'équation est : 0.9999999999093827
```

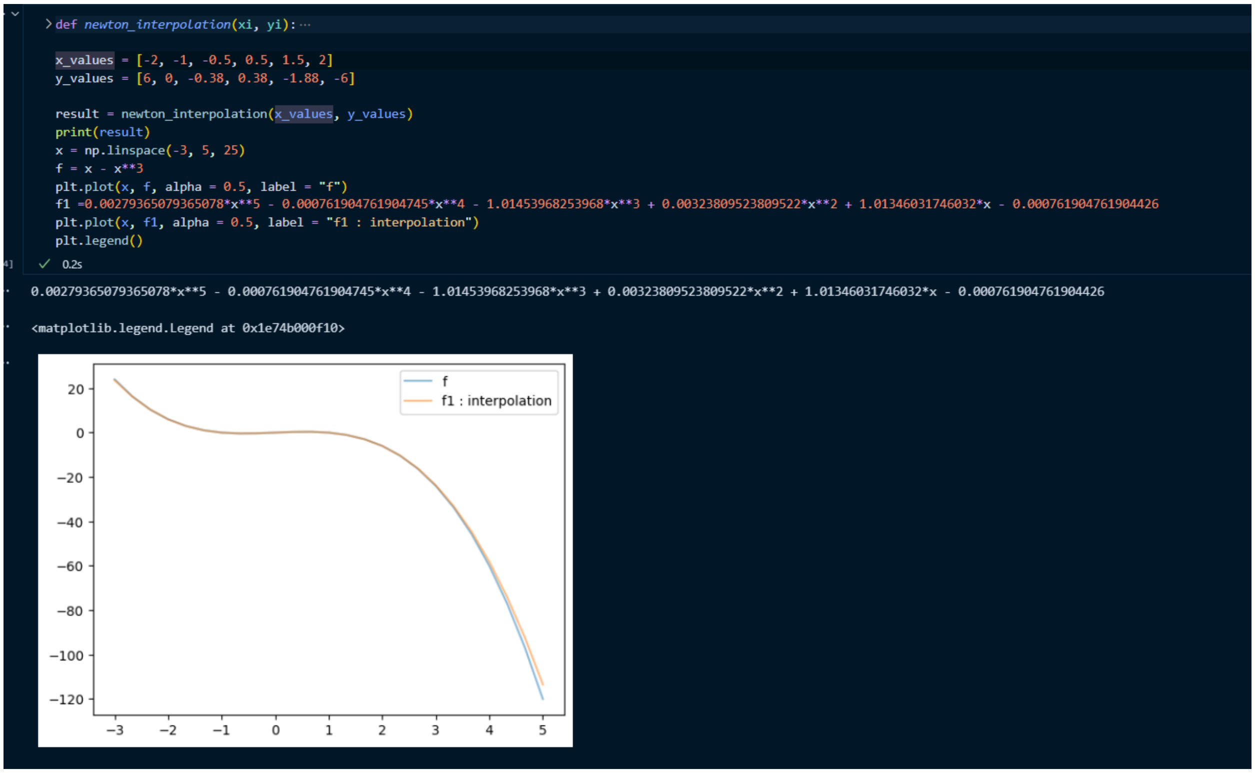
3



```
> def lagrange_interpolation(xi, y):...
  x = symbols('x')
  f_x = x - x**3
  x_values = [-2, -1, -0.5, 0.5, 1.5, 2]
  y_values = [6, 0, -0.38, 0.38, -1.88, -6]
  result = lagrange_interpolation(x_values, y_values)
  print(result)
  x = np.linspace(-4, 5, 30)
  f = x - x**3
  plt.plot(x, f, alpha = 0.5 , label = "f")
  f1 = 0.00279365079365079*x**5 - 0.000761904761904686*x**4 - 1.01453968253968*x**3 + 0.00323809523809571*x**2 + 1.01346031746032*x - 0.000761904761904852
  plt.plot(x, f1, alpha = 0.5, label = "f1")
  plt.legend()
```

```
[3]  ✓ 1.8s
0.00279365079365079*x**5 - 0.000761904761904686*x**4 - 1.01453968253968*x**3 + 0.00323809523809571*x**2 + 1.01346031746032*x - 0.000761904761904852

<matplotlib.legend.Legend at 0x1e74af94c10>
```



```
> def newton_interpolation(xi, yi):...

  x_values = [-2, -1, -0.5, 0.5, 1.5, 2]
  y_values = [6, 0, -0.38, 0.38, -1.88, -6]

  result = newton_interpolation(x_values, y_values)
  print(result)
  x = np.linspace(-3, 5, 25)
  f = x - x**3
  plt.plot(x, f, alpha = 0.5, label = "f")
  f1 =0.00279365079365078*x**5 - 0.000761904761904745*x**4 - 1.01453968253968*x**3 + 0.00323809523809522*x**2 + 1.01346031746032*x - 0.000761904761904426
  plt.plot(x, f1, alpha = 0.5, label = "f1 : interpolation")
  plt.legend()
```

```
✓ 0.2s
0.00279365079365078*x**5 - 0.000761904761904745*x**4 - 1.01453968253968*x**3 + 0.00323809523809522*x**2 + 1.01346031746032*x - 0.000761904761904426

<matplotlib.legend.Legend at 0x1e74b000f10>
```