In [ ]:

```
Nurshanov Dias IT3-2208
lab 6 Spline of the first order
Computational Math
```

In [44]:

```
# Input values
import numpy as np


f = np.log

a, b = 1, 10
N = 10
k = 2
```

In [46]:

```
# a)Code implementation
x_i = np.linspace(a, b, N)
y_i = f(x_i)

x_hat = []
for i in range(len(x_i) - 1):
    x_hat.extend(np.linspace(x_i[i], x_i[i+1], k+2)[1:-1])  # Add intermediary points between nodes
x_hat = np.array(x_hat)

def calc_spline(x_hat, x_i, y_i):
    spline_values = []
    for x in x_hat:
        for i in range(len(x_i) - 1):
            if x_i[i] <= x <= x_i[i+1]:
                # Linear interpolation
                s_x = (y_i[i+1] - y_i[i]) / (x_i[i+1] - x_i[i]) * (x - x_i[i]) + y_i[i]
                spline_values.append(s_x)
                break
    return np.array(spline_values)

spline_values = calc_spline(x_hat, x_i, y_i)

f_values_hat = f(x_hat)

delta = np.abs(f_values_hat - spline_values)
```

In [ ]:

```
```

In [47]:

```
# b)Output should be a table with results
import pandas as pd

df = pd.DataFrame({
    'x': x_hat,
    'f(x)': f_values_hat,
    's(x)': spline_values,
    'delta |f(x) - s(x)|': delta
})
df
```

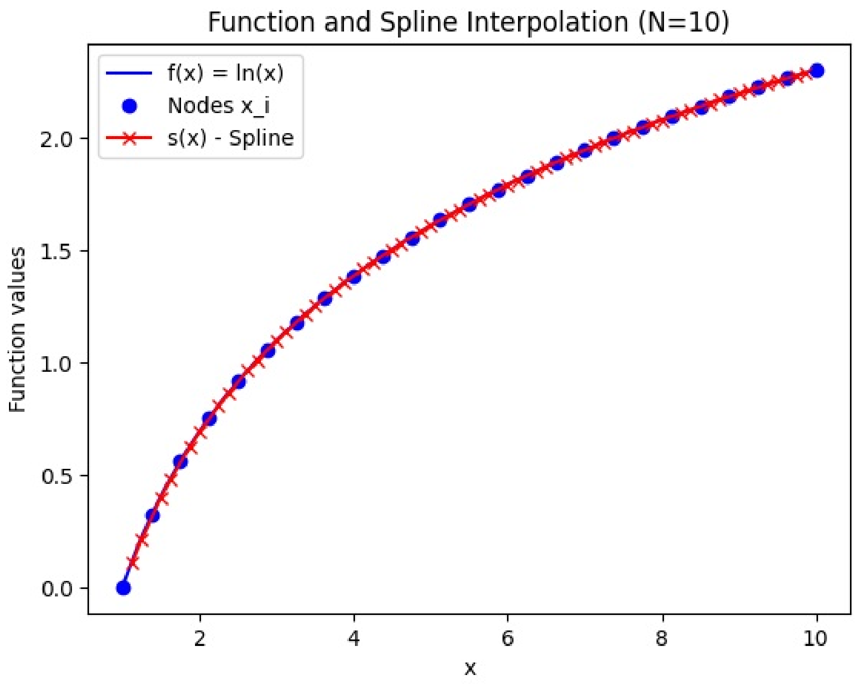
Out[47]:

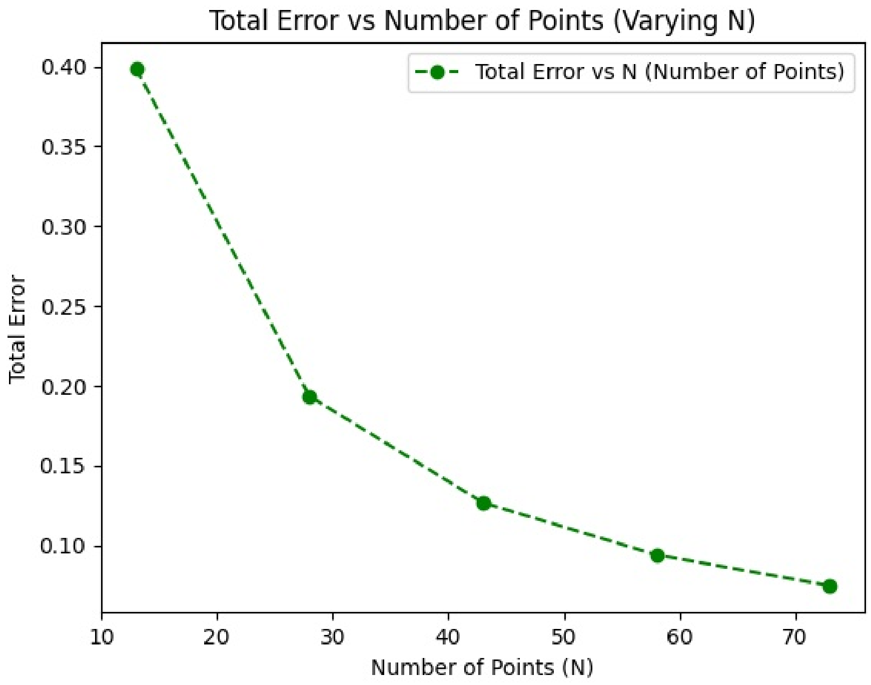
	x	f(x)	s(x)	delta |f(x) - s(x)|
0	1.125	0.117783	0.106151	0.011632
1	1.250	0.223144	0.212302	0.010841
2	1.500	0.405465	0.398841	0.006624
3	1.625	0.485508	0.479228	0.006279
4	1.875	0.628609	0.624334	0.004274
5	2.000	0.693147	0.689053	0.004094
6	2.250	0.810930	0.807945	0.002985
7	2.375	0.864997	0.862118	0.002880
8	2.625	0.965081	0.962878	0.002203
9	2.750	1.011601	1.009465	0.002136
10	3.000	1.098612	1.096920	0.001692
11	3.125	1.139434	1.137788	0.001647
12	3.375	1.216395	1.215055	0.001341
13	3.500	1.252763	1.251455	0.001308
14	3.750	1.321756	1.320668	0.001088
15	3.875	1.354546	1.353481	0.001065
16	4.125	1.417066	1.416165	0.000901
17	4.250	1.446919	1.446036	0.000883
18	4.500	1.504077	1.503319	0.000758
19	4.625	1.531476	1.530732	0.000744
20	4.875	1.584120	1.583473	0.000647
21	5.000	1.609438	1.608802	0.000636
22	5.250	1.658228	1.657670	0.000558
23	5.375	1.681759	1.681209	0.000550
24	5.625	1.727221	1.726734	0.000487
25	5.750	1.749200	1.748720	0.000480
26	6.000	1.791759	1.791331	0.000428
27	6.125	1.812379	1.811956	0.000422
28	6.375	1.852384	1.852004	0.000380
29	6.500	1.871802	1.871427	0.000375
30	6.750	1.909543	1.909204	0.000339
31	6.875	1.927892	1.927557	0.000335
32	7.125	1.963610	1.963305	0.000304
33	7.250	1.981001	1.980701	0.000301
34	7.500	2.014903	2.014628	0.000275
35	7.625	2.031432	2.031161	0.000272
36	7.875	2.063693	2.063444	0.000249
37	8.000	2.079442	2.079195	0.000247
38	8.250	2.110213	2.109986	0.000227
39	8.375	2.125251	2.125026	0.000225
40	8.625	2.154665	2.154457	0.000208
41	8.750	2.169054	2.168848	0.000206
42	9.000	2.197225	2.197033	0.000191
43	9.125	2.211018	2.210828	0.000189
44	9.375	2.238047	2.237870	0.000176
45	9.500	2.251292	2.251117	0.000175
46	9.750	2.277267	2.277104	0.000163
47	9.875	2.290006	2.289845	0.000162

In [48]:

```
# c) 2 graphs: Distribution of values by two functions and the dependence of delta based on the number of point
```

```python
import matplotlib.pyplot as plt

#first graph
x_graph = np.linspace(1, 10, 100)
y_graph = f(x_graph)

plt.figure()
plt.plot(x_graph, y_graph, '-', label='f(x) = ln(x)', color='blue')
plt.plot(x_i, f(x_i), 'bo', label='Nodes x_i')  # Highlight nodes
plt.plot(x_hat, spline_values, 'rx-', label='s(x) - Spline')
plt.title('Function and Spline Interpolation (N=10)')
plt.xlabel('x')
plt.ylabel('Function values')
plt.legend()
plt.show()


#second graph
N_values = [5, 10, 15, 20, 25]

total_errors = []
total_points = []

for N in N_values:
    x_i = np.linspace(a, b, N)
    y_i = f(x_i)

    x_hat = []
    for i in range(len(x_i) - 1):
        x_hat.extend(np.linspace(x_i[i], x_i[i+1], k+2)[1:-1])  # Add intermediary points between nodes
    x_hat = np.array(x_hat)

    spline_values = calc_spline(x_hat, x_i, y_i)

    f_values_hat = f(x_hat)

    delta = np.abs(f_values_hat - spline_values)

    total_error = np.sum(delta)
    total_errors.append(total_error)

    total_points.append(len(x_i) + len(x_hat))

plt.figure()
plt.plot(total_points, total_errors, 'g--o', label='Total Error vs N (Number of Points)')
plt.title('Total Error vs Number of Points (Varying N)')
plt.xlabel('Number of Points (N)')
plt.ylabel('Total Error')
plt.legend()
plt.show()
```





In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js