Assignment 3

The lossless Microstrip Line

ELG 3106 - Electromagnetic Engineering

Fall 2023

School of Electrical Engineering and Computer Science

University of Ottawa

Tristan Ruel 300272156

November 13th, 2023

Index

Executive Summary

The following assignment focuses on the task of computing the effective relative permittivity ($\mathcal{E}_{eff}$) and the characteristic impedance (Z_0) of a microstrip line. Utilizing Python for automation, this study involved calculating $\mathcal{E}_{eff}$ and Z_0, then reverse-engineering the width-to-thickness ratio (S) from Z_0 using two approximation methods, *a* and *b*. The accuracy of these methods was evaluated by comparing the calculated S values with those obtained from the Module 2.3 app.

The calculations were conducted for different values of relative permittivity ($\mathcal{E}_r$), and the results were documented and analyzed. The assignment presents detailed tabulated results and graphical representations, highlighting the relative differences between the calculated and provided Z_0 values, and illustrating the percent differences in Z_0 for varying values of $\mathcal{E}_r$.

The discussion section delves into the complexities and challenges of the calculations, revealing that although the approximation method yielded values close to the exact ones, it was still intricate in nature. The study showed that approximation (a) tended to deviate significantly with higher characteristic impedance, surpassing a 2% difference threshold, especially for $\mathcal{E}_r$=10 and $\mathcal{E}_r$=2. In contrast, approximation (b) maintained a closer alignment with the original S values, proving more reliable within specific impedance ranges.

In conclusion, while the approximation methods offered a streamlined approach to determining microstrip line parameters, they also exhibited inherent limitations which are detailed further in this assignment.

Introduction

The core of the following assignment is to compute the effective relative permittivity ($\mathcal{E}_{eff}$) and the characteristic impedance (Z_0) of a microstrip line. These parameters are crucial for the accurate design and analysis of microwave circuits. The effective permittivity is calculated through a function that considers the relative permittivity of the dielectric ($\mathcal{E}_r$) and the width-to-thickness ratio (s), while the characteristic impedance is derived from a logarithmic expression that incorporates $\mathcal{E}_{eff}$.

Moving forward, we will reverse-engineer the width-to-thickness ratio from a given characteristic impedance using two different approximations. We will then evaluate the accuracy of these approximations by comparing the calculated S values with those obtained from the specialized Module 2.3 app (F. Ulaby, 2020).

We are tasked with automating these calculations, producing plots that illustrate the relative differences between the calculated and provided Z_0 values, and determining the validity range of the approximations. For this assignment, we will be using Python to automate these computations.

Theory

Although microstrip transmission lines exhibit complex properties, employing introductory approximations and a straightforward method can yield reasonably accurate outcomes. Instead of determining the exact relative permittivity $\mathcal{E}_{eff}$ such that $u_p = \frac{c}{\sqrt{\varepsilon_{eff}}}$, it's feasible to use an estimated permittivity value as shown here:

$$\varepsilon_{eff} = \frac{\varepsilon_r+1}{2} + \left(\frac{\varepsilon_r-1}{2}\right)\left(1 + \frac{10}{s}\right)^{-xy}$$

Where $s = \frac{w}{h}$ the width-to-thickness ratio and x and y are given by:

$$x = 0.56\left(\frac{\varepsilon_r-0.9}{\varepsilon_r+3}\right)$$

$$y = 1 + 0.02ln\left(\frac{s^4+0.00037s^2}{s^4+0.43}\right) + 0.05ln(1 + 0.00017s^3)$$

The characteristic impedance is given by:

$$Z_0 = \frac{60}{\sqrt{\varepsilon_{eff}}}ln\left(\frac{6+(2\pi-6)e^{-t}}{s} + \sqrt{1 + \frac{4}{s^2}}\right) \text{ where } t = \left(\frac{30.67}{s}\right)^{0.75}$$

Hence, for given values of $\mathcal{E}_r$, h and w, we can calculate Z_0

The following provided expressions are used to approximate the width-to-thickness ratios for certain ranges of Z_0. These expressions are said to be accurate to determine s within 2%.

a) For $Z_0 \leq (44-2\mathcal{E}_r)\ \Omega$

$$s = \frac{w}{h} = \frac{2}{\pi}\left((q - 1) - ln(2q - 1) + \frac{\varepsilon_r-1}{\varepsilon_r+1}\left[ln(q - 1) + 0.29 - \frac{0.59}{\varepsilon_r}\right]\right) \text{where } q = \frac{60\pi^2}{Z_0\sqrt{\varepsilon_r}}$$

b) For $Z_0 \geq (44-2\mathcal{E}_r)\ \Omega$

$$s = \frac{w}{h} = \frac{8e^p}{e^{2p}-2} \text{ where } p = \sqrt{\frac{\varepsilon_r+1}{2}}\left(\frac{Z_0}{60}\right) + \left(\frac{\varepsilon_r-1}{\varepsilon_r+1}\right)\left(0.23 + \frac{0.12}{\varepsilon_r}\right)$$

*The entirety of this section was created using the works of Dr. H. Schrimer. (Schriemer, 2023)

Flow Chart

- **Start**
- **Import Libraries** (NumPy, matplotlib.pyplot, Pandas)
- **Initialize Constants** (c = 3e8)
- **Define Functions**
  - calculate_x(Er)
  - calculate_y(s)
  - calculate_Eeff(Er, s, x, y)
  - calculate_Z0(Eeff, s)
  - approximation_a(Z0, Er)
  - approximation_b(Z0, Er)
- **Set Parameters for Er=10**
  - Initialize Er_10, h, s_values_10, provided_Z0_values_10
- **Calculating Z0 Values for Er=2:**
  - Define Er_2, s_values_2, and an empty list Z0_values_2.
  - For each s in s_values_2:
    - Calculate x, y, Eeff, and Z0 using the defined functions.
    - Append Z0 to Z0_values_2.
- **Reverse Engineering S values:**
  - For Er=10, calculate reverse_engineered_s_values_a_10 and reverse_engineered_s_values_b_10 using approximation_a and approximation_b.
  - For Er=2, calculate reverse_engineered_s_values_a_2 and reverse_engineered_s_values_b_2.
- **Calculating Relative Differences for Er=10:**
  - Compute relative_differences_a_10 and relative_differences_b_10.
- **Plotting Relative Differences for Er=10:**
  - Plot and save relative_differences_a_10 and relative_differences_b_10 against provided_Z0_values_10.
- **Calculating Relative Differences for Er=2:**
  - Compute relative_differences_a_2 and relative_differences_b_2.
- **Plotting Relative Differences for Er=2:**
  - Plot and display relative_differences_a_2 and relative_differences_b_2 against Z0_values_2.
- **Calculating and Plotting Percent Difference in Z0 for Er=10:**
  - Compute calculated_Z0_values_10 and percent_diff_Z0_10.
  - Plot and save percent_diff_Z0_10 against s_values_10.
- **Determining Valid Ranges:**
  - Determine valid_range_a_10, valid_range_b_10, valid_range_a_2, valid_range_b_2 based on relative differences.

- **Printing Valid Ranges:**
  - Print valid ranges for both approximations A and B, for both Er=10 and Er=2.
- **Saving Data to Excel:**
  - Create a DataFrame df from valid ranges.
  - Save df to an Excel file.
- **Display Plots:**
  - Display all generated plots.
- **End**

Tabulated Results

S	***Z0***$(\mathcal{E}_r=10)$	***Z0***$(\mathcal{E}_r=2)$	***S***$(\mathcal{E}_r=10)$ **(a)**	***S***$(\mathcal{E}_r=10)$ **(b)**	***S***$(\mathcal{E}_r=2)$ **(a)**	***S***$(\mathcal{E}_r=2)$ **(b)**
0.5	65.922149	131.474566	0.432098	0.502865	0.450086	0.499965
0.6	61.353976	122.713307	0.542833	0.603199	0.558849	0.599871
0.7	57.522488	115.359366	0.650526	0.703412	0.665213	0.699754
0.8	54.233351	109.041255	0.756203	0.803502	0.769947	0.799625
0.9	51.361762	103.519564	0.860489	0.903443	0.873540	0.899484
1.0	48.822521	98.630590	0.963776	1.003193	0.976307	0.999327
1.1	46.554488	94.257010	1.066317	1.102715	1.078462	1.099155
1.2	44.512023	90.311550	1.168287	1.201986	1.180153	1.198972
1.3	42.660015	86.727351	1.269809	1.300994	1.281484	1.298794

1.4	40.970819	83.451977	1.370974	1.399742	1.382532	1.398642
1.5	39.422266	80.443503	1.471850	1.498241	1.483351	1.498542
1.6	37.996303	77.667868	1.572487	1.596510	1.583982	1.598525
1.7	36.678036	75.097011	1.672921	1.694569	1.684453	1.698623
1.8	35.455039	72.707539	1.773181	1.792445	1.784782	1.798870
1.9	34.316841	70.479742	1.873286	1.890161	1.884984	1.899299
2.0	33.254538	68.396860	1.973248	1.987744	1.985064	1.999945
2.1	32.260498	66.444521	2.073075	2.085220	2.085026	2.100842
2.2	31.328132	64.610308	2.172772	2.182613	2.184869	2.202022
2.3	30.451715	62.883422	2.272340	2.279948	2.284590	2.303517
2.4	29.626241	61.254414	2.371776	2.377246	2.384185	2.405359

2.5	28.847309	59.714968	2.471078	2.474530	2.483646	2.507576
2.6	28.111032	58.257728	2.570240	2.571820	2.582966	2.610199
2.7	27.413960	56.876159	2.669255	2.669134	2.682136	2.713254
2.8	26.753017	55.564429	2.768117	2.766490	2.781148	2.816769
2.9	26.125451	54.317312	2.866818	2.863906	2.879991	2.920768
3.0	25.528791	53.130109	2.965349	2.961395	2.978657	3.025277
3.1	24.960809	51.998579	3.063701	3.058974	3.077134	3.130320
3.2	24.419492	50.918884	3.161867	3.156654	3.175413	3.235919
3.3	23.903013	49.887538	3.259836	3.254449	3.273485	3.342096
3.4	23.409713	48.901369	3.357601	3.352370	3.371339	3.448874
3.5	22.938077	47.957481	3.455153	3.450429	3.468968	3.556273
3.6	22.486724	47.053223	3.552483	3.548634	3.566360	3.664314

4.4	19.473430	40.990975	4.322231	4.340621	4.336077	4.554036
4.5	19.158138	40.353982	4.417240	4.440525	4.431012	4.668703
4.6	18.854091	39.739202	4.511960	4.540651	4.525645	4.784197
4.7	18.560707	39.145505	4.606389	4.641006	4.619971	4.900536

4.8	18.277444	38.571836	4.700522	4.741596	4.713985	5.017738
4.9	18.003792	38.017206	4.794355	4.842426	4.807685	5.135820
5.0	17.739277	37.480691	4.887884	4.943502	4.901068	5.254802

Table 1.0 - Calculated Data

Graphical Results

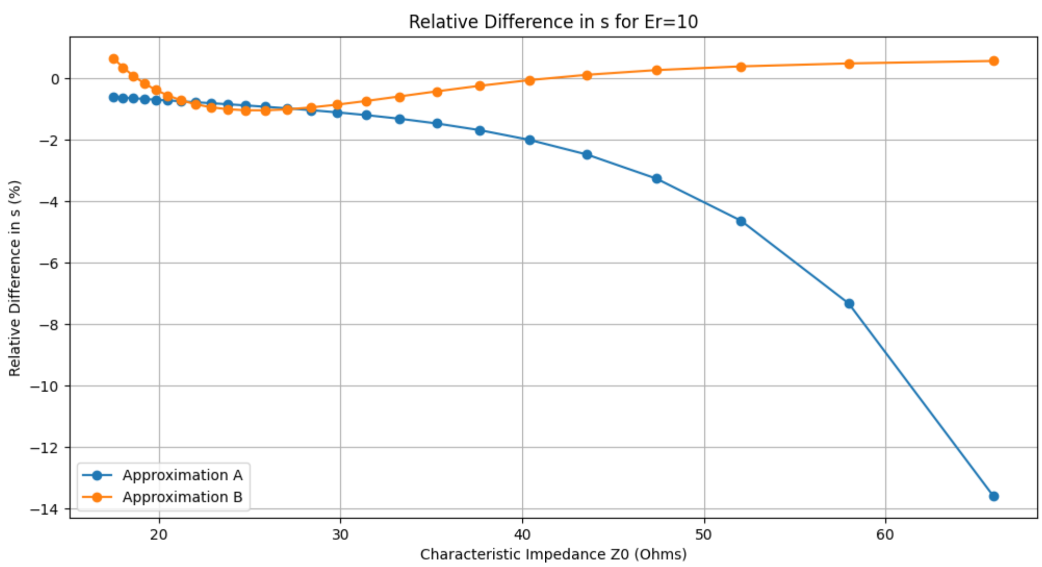

Figure 1.0 - Relative Difference in s for E_r=10

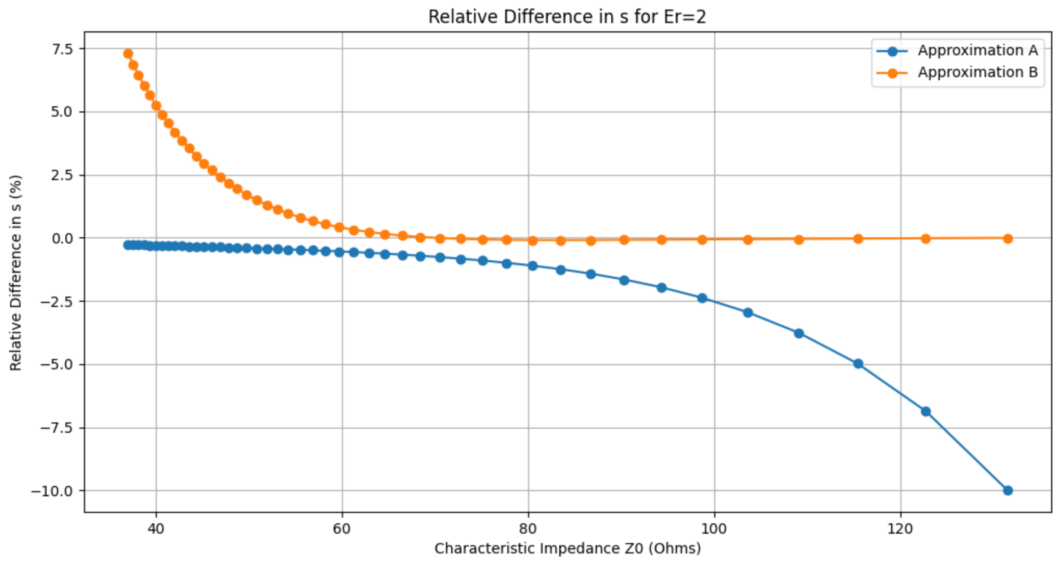

Figure 1.1 - Relative Difference in s for E_r=2

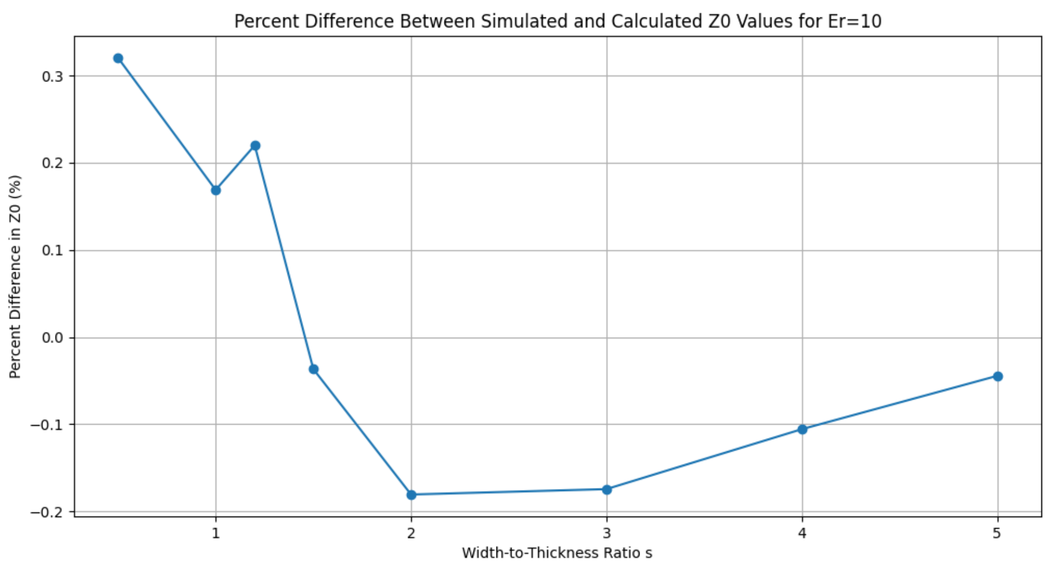

Figure 1.2 - Percent Difference Between Simulated and Calculated Z0 Values for E_r=10

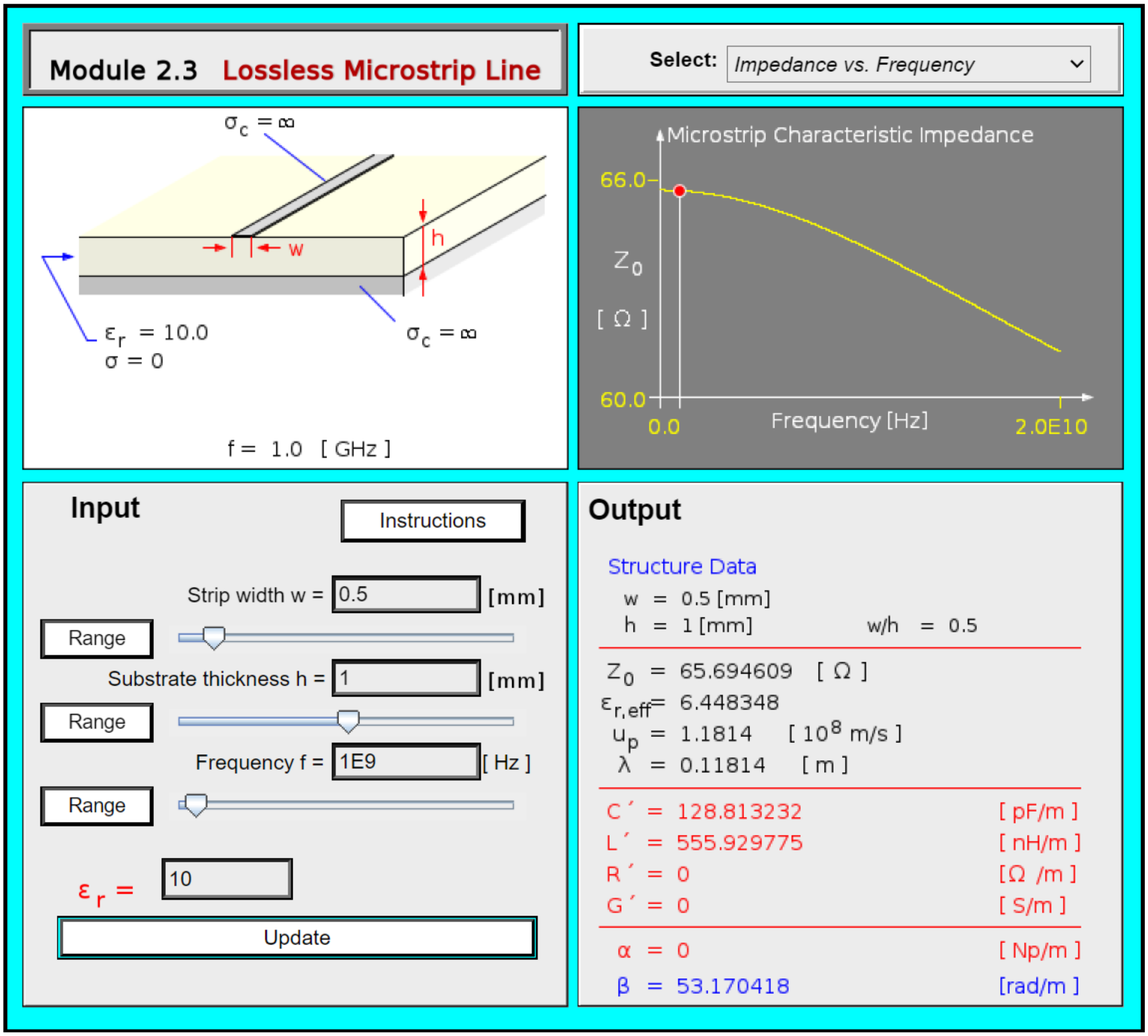

Figure 2.0 - Screenshot of Module 2.3 App for S=0.5

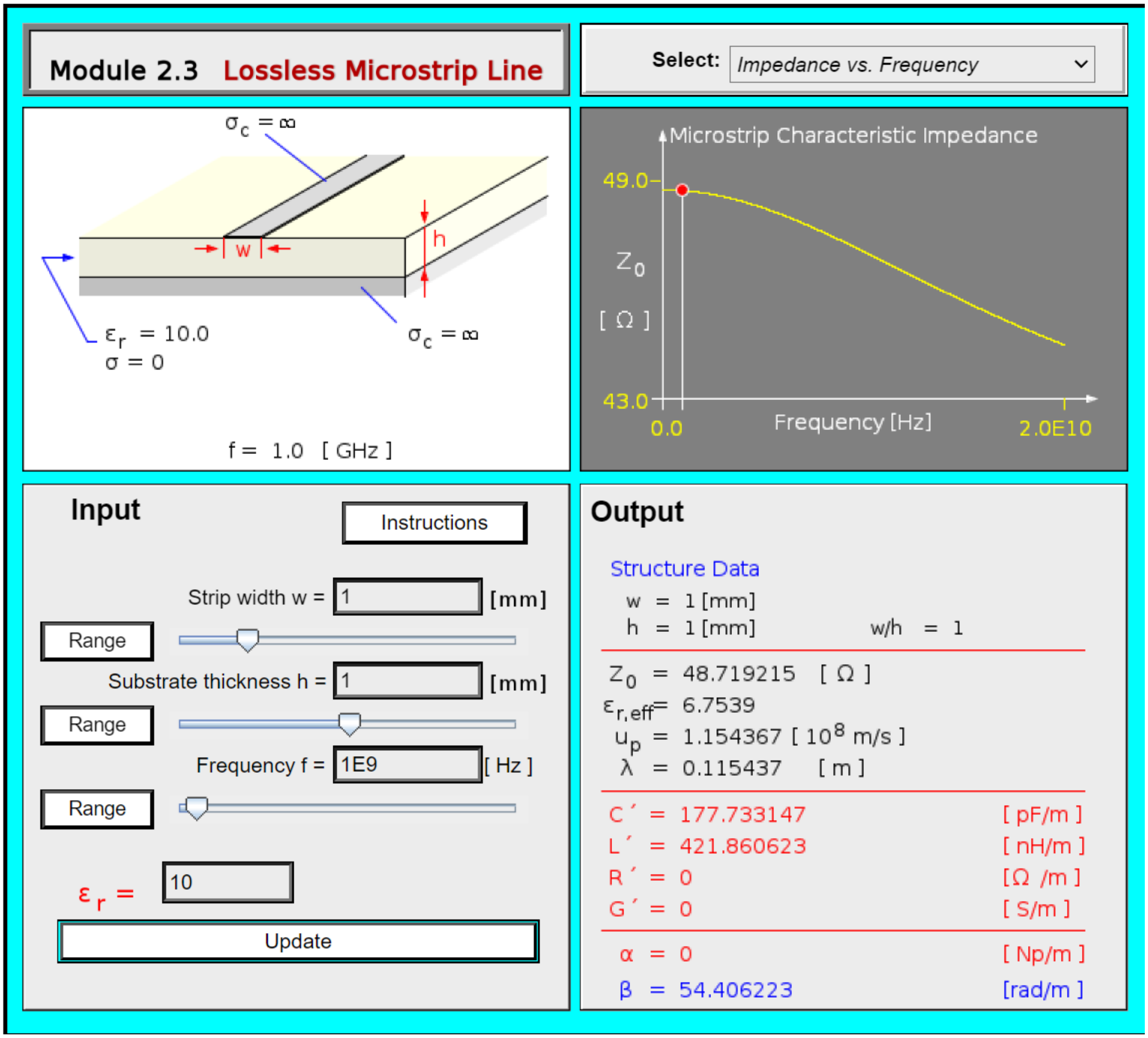

Figure 2.1 - Screenshot of Module 2.3 App for S=1.0

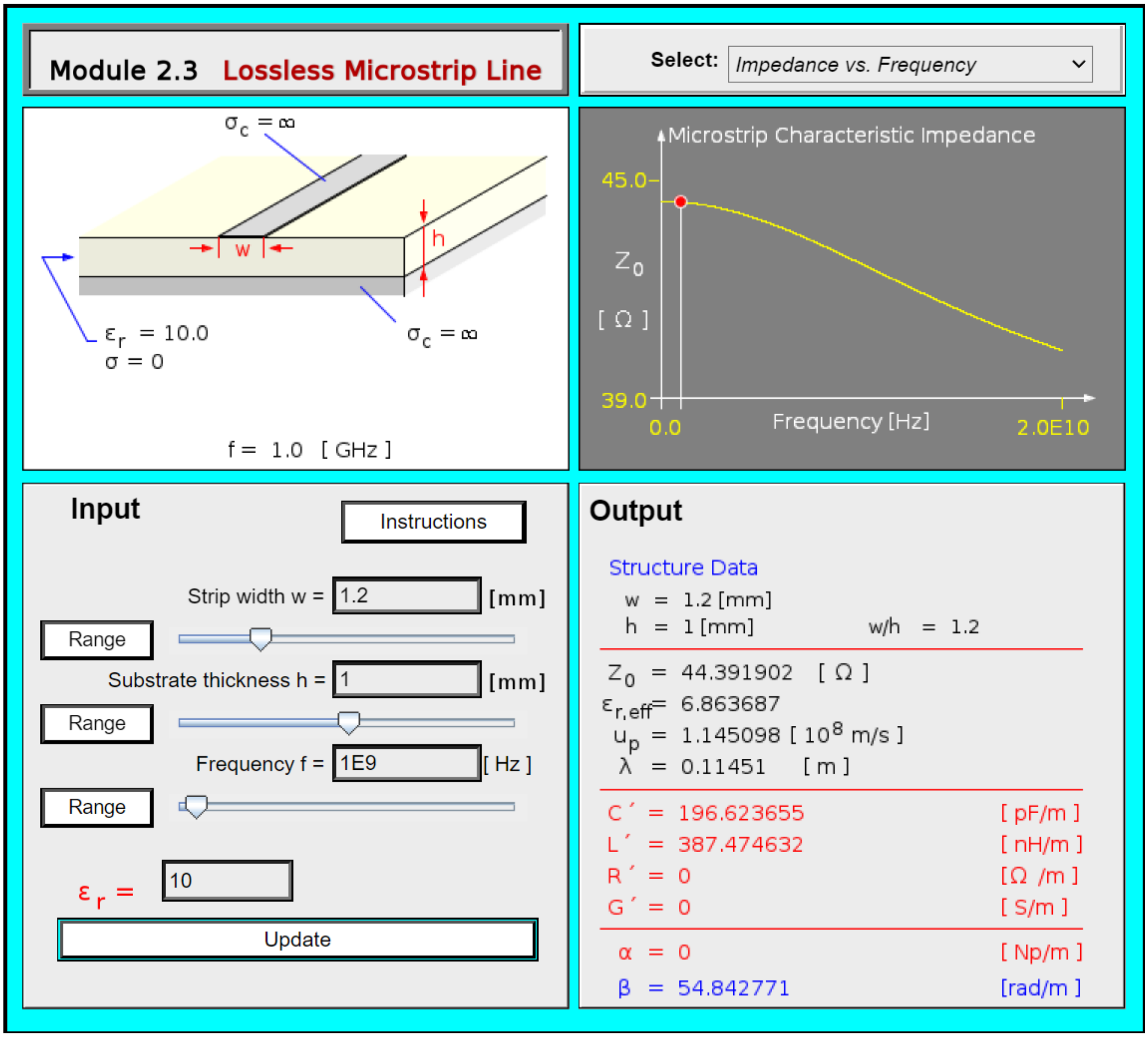

Figure 2.2 - Screenshot of Module 2.3 App for S=1.2

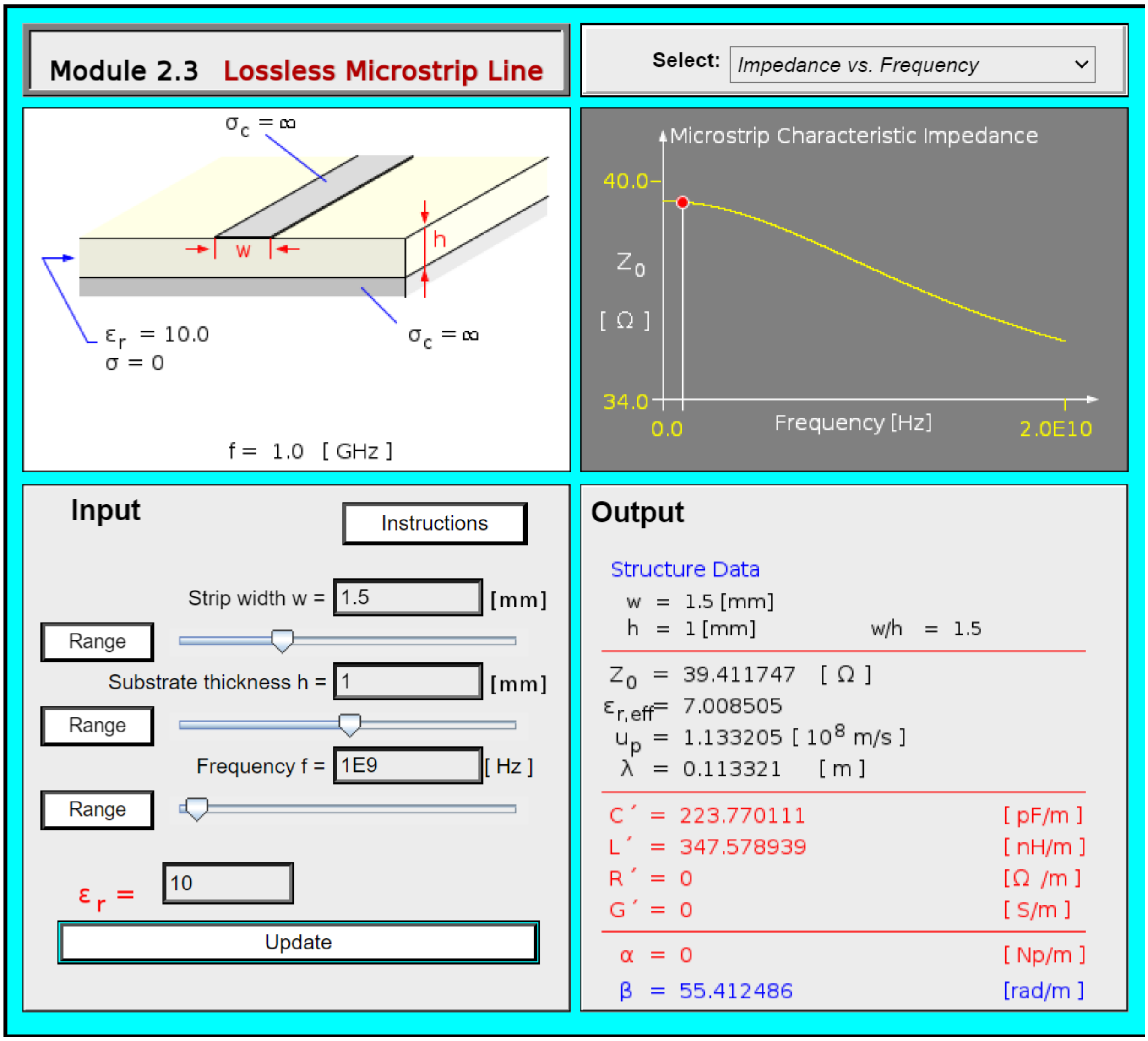

Figure 2.3 - Screenshot of Module 2.3 App for S=1.5

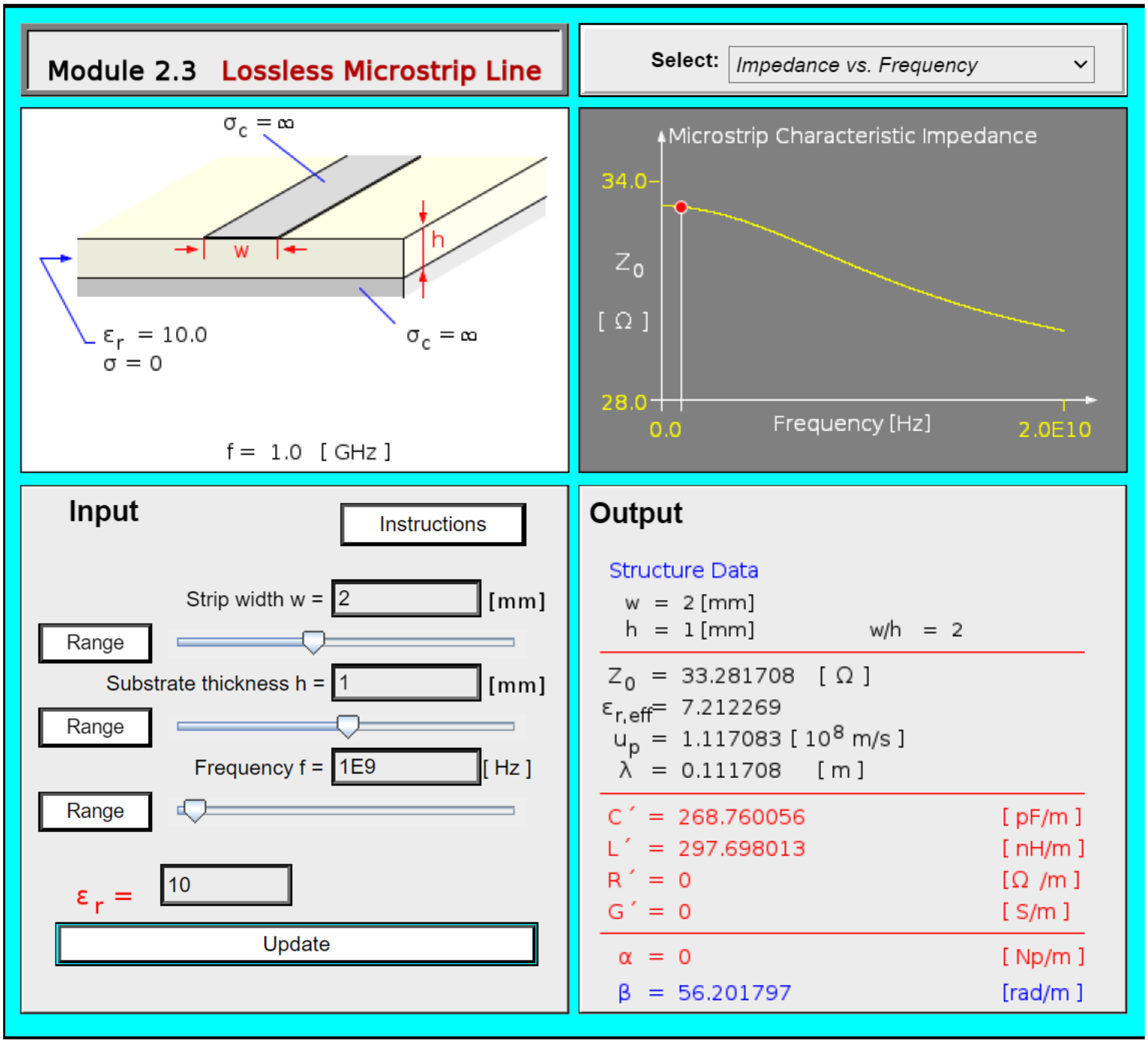

Figure 2.4 - Screenshot of Module 2.3 App for S=2.0

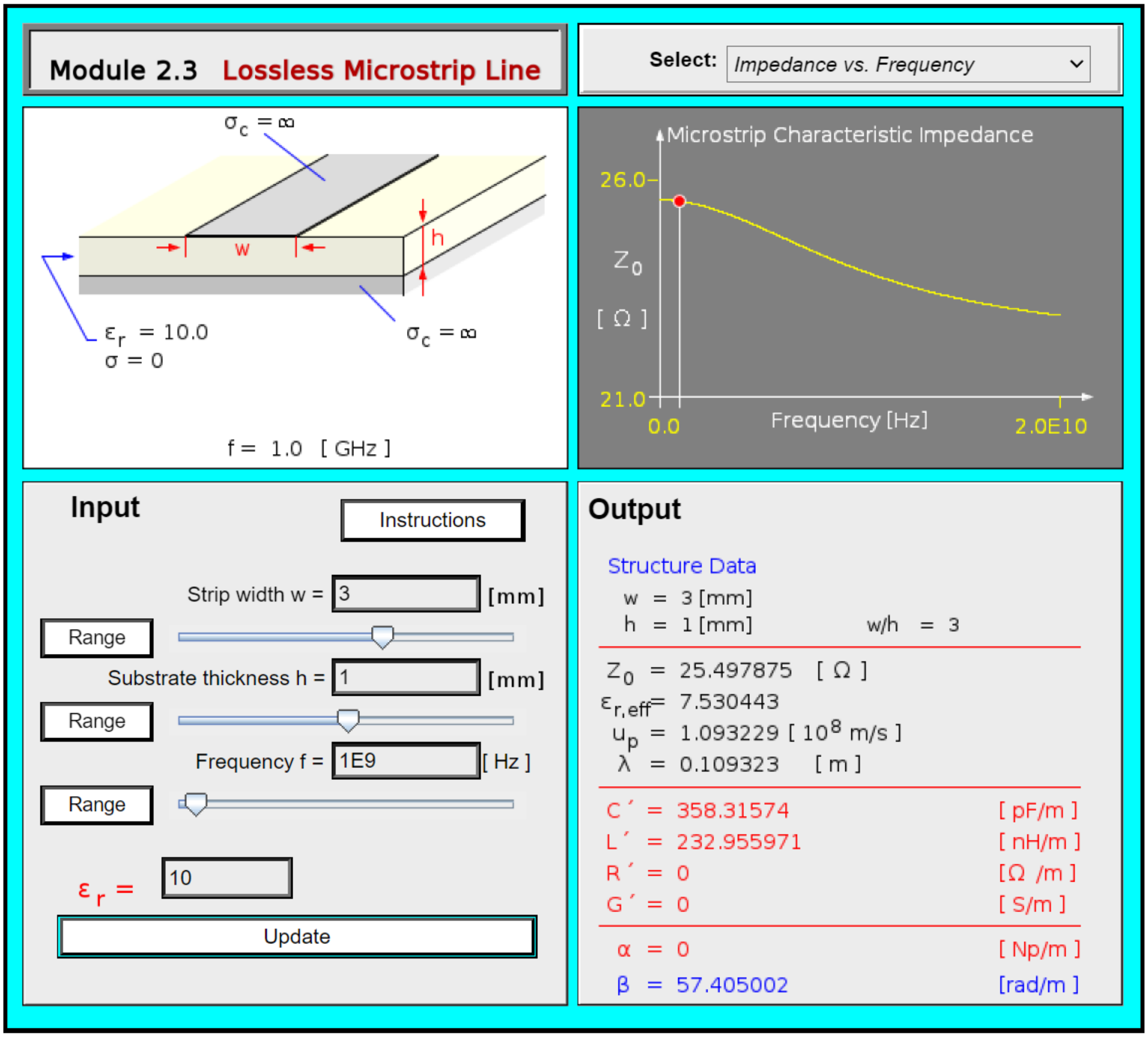

Figure 2.5 - Screenshot of Module 2.3 App for S=3

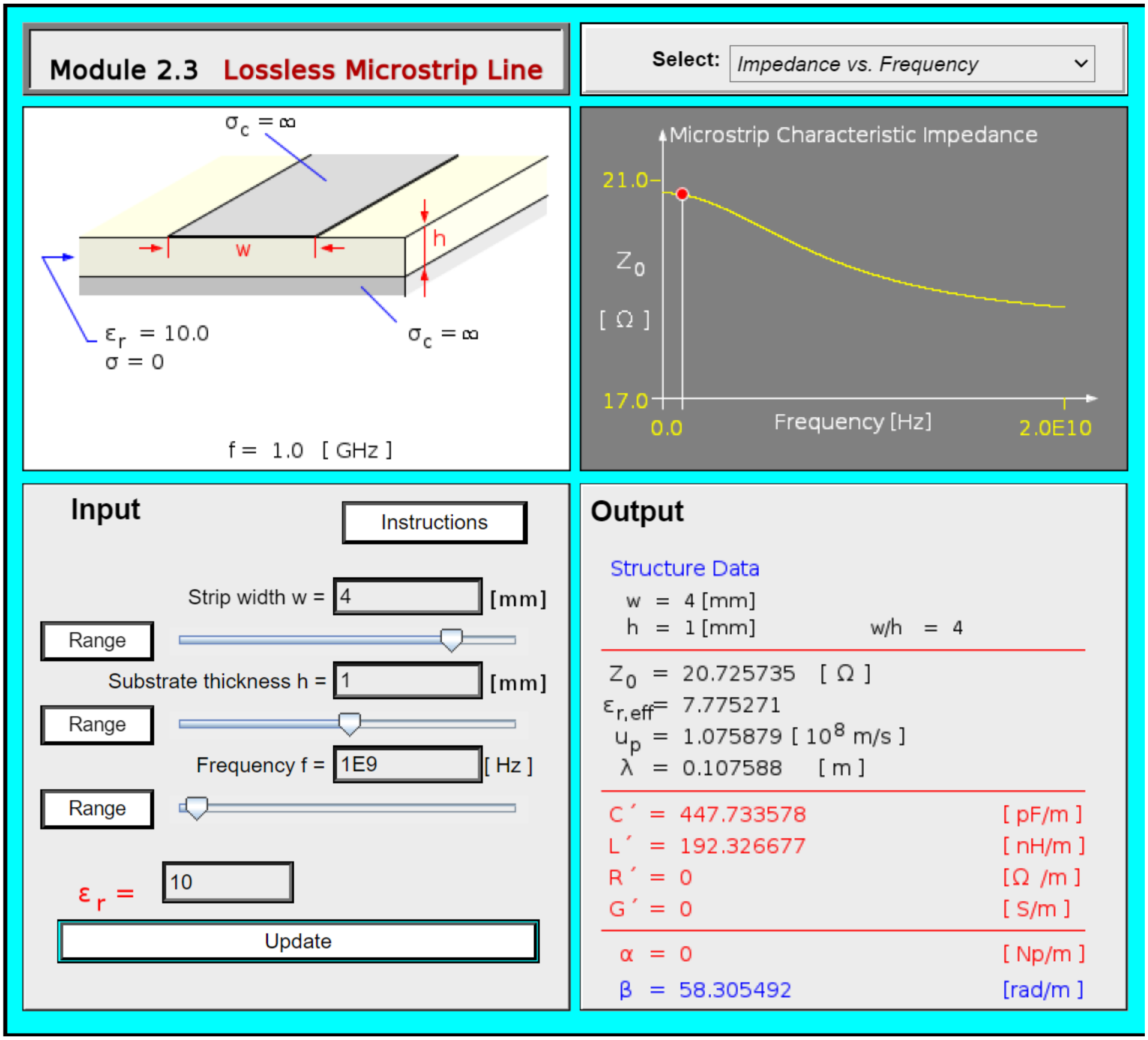

Figure 2.6 - Screenshot of Module 2.3 App for S=4

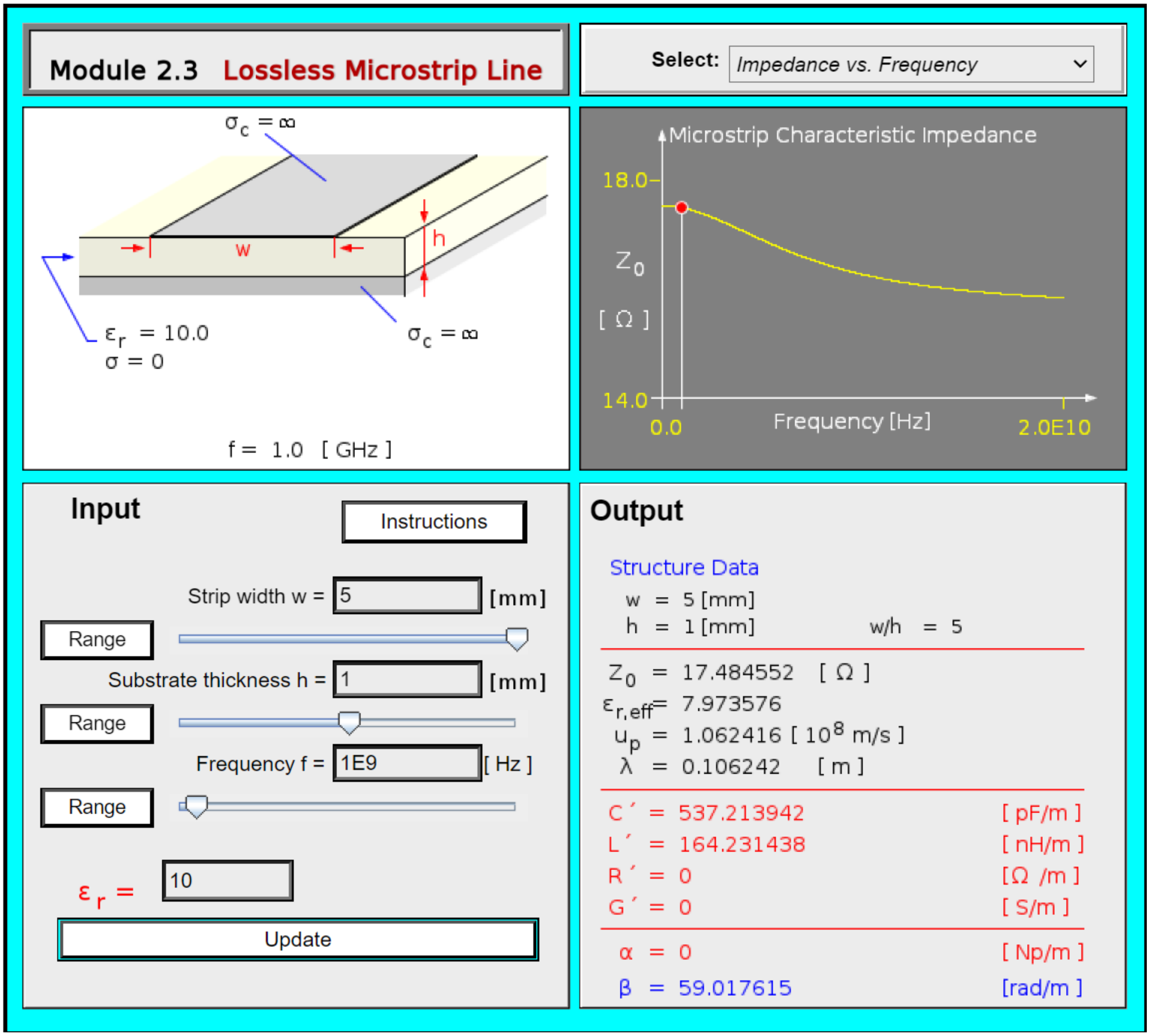

Figure 2.7 - Screenshot of Module 2.3 App for S=5

*This section was created using the works of the Module 2.3 app (F. Ulaby, 2020)

Discussion

The approximation methods produced values close to the exact ones, but was itself somewhat complex, highlighting the challenges in precise calculations. Initially, the impedance values Z_0 for S ranging from 0.5 to 5 were computed, as shown in *Table 1.0* under the $Z0(\mathcal{E}_r\!=\!10)$ column, using a relative permittivity of $\mathcal{E}_r$=10.

Subsequently, for S values of 0.5, 1.0, 1.2, 1.5, 2.0, 3.0, 4.0, and 5.0, Z_0 was simulated using the Module 2.3 app, and the outcomes were documented through screenshots (*Figures 2.0-2.7*). These S values were also used for direct Z_0 calculations, and the results were compared with the simulated values. A percent difference analysis of these two data sets was graphed in *Figure 1.2*.

The process was repeated for Z_0 calculations over the same S range, but with a relative permittivity of $\mathcal{E}_r$=2, and results were noted in *Table 1.0* under the Z_0 ($\mathcal{E}_r$=2) column. Using both $\mathcal{E}_r$=10 and $\mathcal{E}_r$=2 values, the corresponding S values were calculated using two approximation methods (a) and (b), and each was compared to the original S values. The percent differences were plotted in two graphs, presented in *Figure 1.0* and *Figure 1.1*, for $\mathcal{E}_r$=10 and $\mathcal{E}_r$=2, respectively.

Figure 1.0 and *Figure 1.1* reveal that certain segments of the approximation are more accurate than others. In both scenarios, the absolute value of the percent difference in approximation (a) grows exponentially with increasing characteristic impedance. Approximation (a) exceeds the 2% difference threshold after an impedance of approximately 40Ω for $\mathcal{E}_r$=10, and around 95Ω for $\mathcal{E}_r$=2. Meanwhile, approximation (b) remains closely aligned with the original S

values, staying within a 1% difference for $\mathcal{E}_r$=10 and only deviating significantly when the characteristic impedance drops below approximately 50Ω for $\mathcal{E}_r$=2.

Conclusion

The assignment's comprehensive analysis of microstrip line parameters for effective relative permittivity ($\mathcal{E}_{eff}$) and characteristic impedance (Z_0), has demonstrated the effectiveness and limitations of approximation methods in reverse-engineering the width-to-thickness ratio (S) from Z_0 values. By automating calculations using Python and comparing results with the Module 2.3 app, it became evident that while approximation methods offer a streamlined approach, they come with inherent complexities and limitations.

The calculated values for $\mathcal{E}_{eff}$ and Z_0 using the provided formulas and the reverse-engineered S values using approximations (a) and (b) underscored the intricate relationship between these parameters. The graphical representation of the relative differences and the percent difference analysis provided a clear visualization of the effectiveness of each approximation method. Notably, approximation (a) showed increasing deviation from actual values with higher characteristic impedance, especially beyond the 2% difference threshold for both $\mathcal{E}_r$=10 and $\mathcal{E}_r$=2. On the other hand, approximation (b) maintained a closer alignment with the original S values, demonstrating its reliability within higher impedance ranges.

This assignment has not only affirmed the viability of approximation methods in certain contexts but also highlighted their limitations, particularly in scenarios of high characteristic impedance. The insights gained from this investigation are invaluable for the design and analysis of microwave circuits, providing a foundation for selecting appropriate methods based on specific requirements and constraints.

References

- H. Schriemer, "Assignment 3," University of Ottawa, Ottawa, Ontario, Canada, Nov. 7, 2023
- F. Ulaby, "Module 2.3." University of Michigan, Dept. of EECS. [Online]. Available: https://em8e.eecs.umich.edu/jsmodules/ch2/mod2_3.html. Accessed on: Nov. 7, 2023.

Appendix
Tables

S	Z0(ε_r=10)	Z0(ε_r=2)	S(ε_r=10) (a)	S(ε_r=10) (b)	S(ε_r=2) (a)	S(ε_r=2) (b)
0.5	65.922149	131.474566	0.432098	0.502865	0.450086	0.499965
0.6	61.353976	122.713307	0.542833	0.603199	0.558849	0.599871
0.7	57.522488	115.359366	0.650526	0.703412	0.665213	0.699754
0.8	54.233351	109.041255	0.756203	0.803502	0.769947	0.799625
0.9	51.361762	103.519564	0.860489	0.903443	0.873540	0.899484
1.0	48.822521	98.630590	0.963776	1.003193	0.976307	0.999327
1.1	46.554488	94.257010	1.066317	1.102715	1.078462	1.099155
1.2	44.512023	90.311550	1.168287	1.201986	1.180153	1.198972
1.3	42.660015	86.727351	1.269809	1.300994	1.281484	1.298794

1.4	40.970819	83.451977	1.370974	1.399742	1.382532	1.398642
1.5	39.422266	80.443503	1.471850	1.498241	1.483351	1.498542
1.6	37.996303	77.667868	1.572487	1.596510	1.583982	1.598525
1.7	36.678036	75.097011	1.672921	1.694569	1.684453	1.698623
1.8	35.455039	72.707539	1.773181	1.792445	1.784782	1.798870
1.9	34.316841	70.479742	1.873286	1.890161	1.884984	1.899299
2.0	33.254538	68.396860	1.973248	1.987744	1.985064	1.999945
2.1	32.260498	66.444521	2.073075	2.085220	2.085026	2.100842
2.2	31.328132	64.610308	2.172772	2.182613	2.184869	2.202022
2.3	30.451715	62.883422	2.272340	2.279948	2.284590	2.303517
2.4	29.626241	61.254414	2.371776	2.377246	2.384185	2.405359

2.5	28.847309	59.714968	2.471078	2.474530	2.483646	2.507576
2.6	28.111032	58.257728	2.570240	2.571820	2.582966	2.610199
2.7	27.413960	56.876159	2.669255	2.669134	2.682136	2.713254
2.8	26.753017	55.564429	2.768117	2.766490	2.781148	2.816769
2.9	26.125451	54.317312	2.866818	2.863906	2.879991	2.920768
3.0	25.528791	53.130109	2.965349	2.961395	2.978657	3.025277
3.1	24.960809	51.998579	3.063701	3.058974	3.077134	3.130320
3.2	24.419492	50.918884	3.161867	3.156654	3.175413	3.235919
3.3	23.903013	49.887538	3.259836	3.254449	3.273485	3.342096
3.4	23.409713	48.901369	3.357601	3.352370	3.371339	3.448874
3.5	22.938077	47.957481	3.455153	3.450429	3.468968	3.556273
3.6	22.486724	47.053223	3.552483	3.548634	3.566360	3.664314

3.7	22.054386	46.186166	3.649583	3.646997	3.663509	3.773015
3.8	21.639900	45.354076	3.746445	3.745525	3.760405	3.882398
3.9	21.242197	44.554897	3.843063	3.844228	3.857041	3.992480
4.0	20.860291	43.786729	3.939427	3.943113	3.953410	4.103281
4.1	20.493272	43.047820	4.035533	4.042187	4.049504	4.214818
4.2	20.140300	42.336544	4.131373	4.141459	4.145317	4.327111
4.3	19.800593	41.651396	4.226941	4.240935	4.240844	4.440178
4.4	19.473430	40.990975	4.322231	4.340621	4.336077	4.554036
4.5	19.158138	40.353982	4.417240	4.440525	4.431012	4.668703
4.6	18.854091	39.739202	4.511960	4.540651	4.525645	4.784197
4.7	18.560707	39.145505	4.606389	4.641006	4.619971	4.900536

4.8	18.277444	38.571836	4.700522	4.741596	4.713985	5.017738
4.9	18.003792	38.017206	4.794355	4.842426	4.807685	5.135820
5.0	17.739277	37.480691	4.887884	4.943502	4.901068	5.254802

Table 1.0 - Calculated Data

Figures

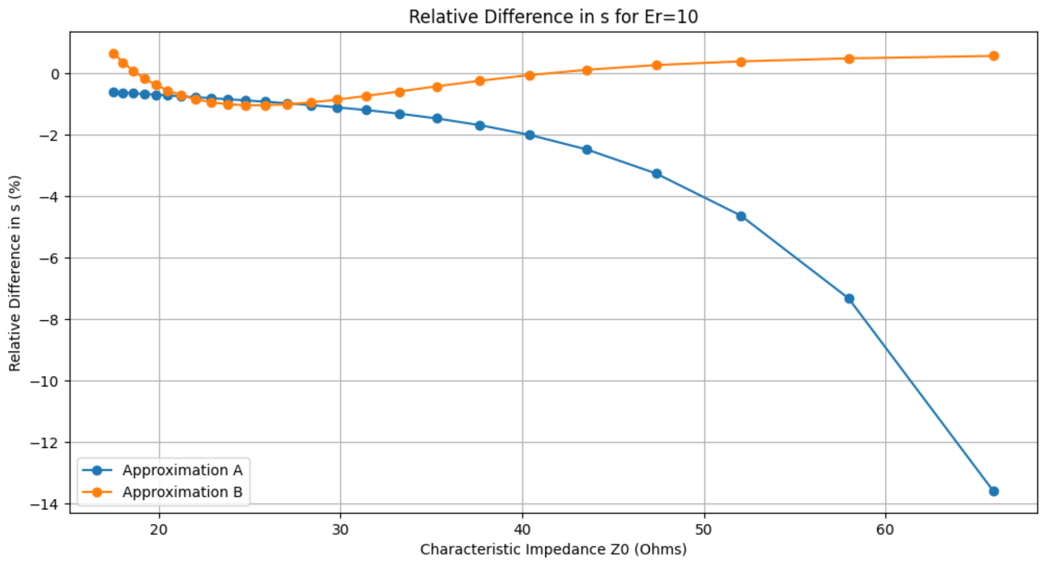

Figure 1.0 - Relative Difference in s for E_r=10

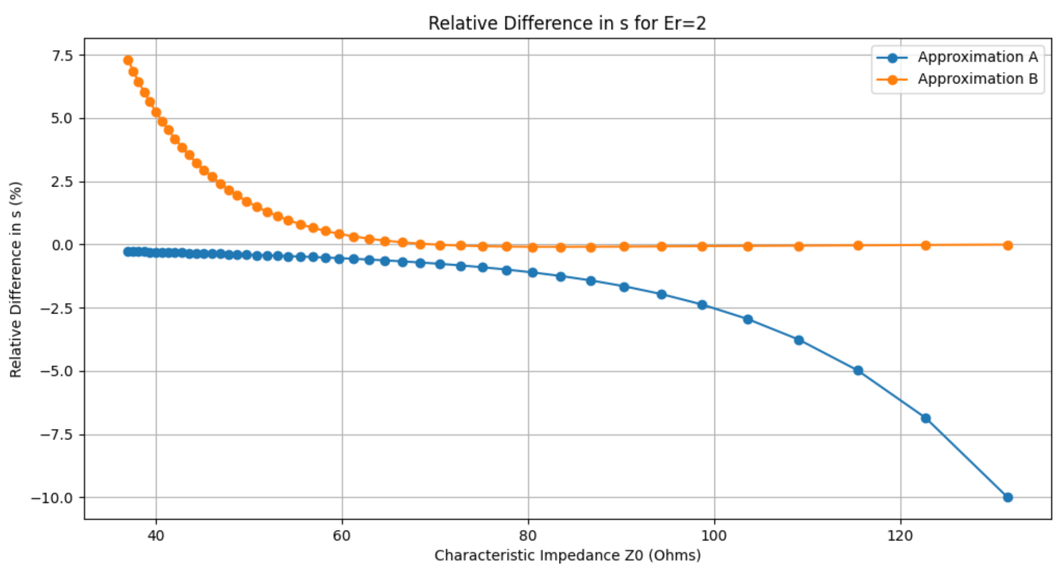

Figure 1.1 - Relative Difference in s for E_r=2

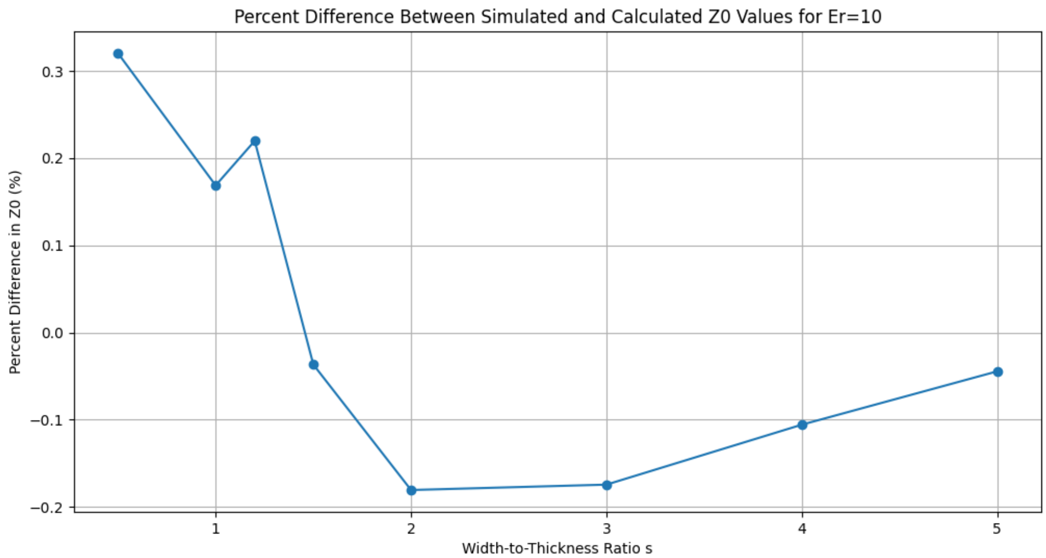

Figure 1.2 - Percent Difference Between Simulated and Calculated Z0 Values for E_r=10

Figure 1.2 - Percent Difference Between Simulated and Calculated Z0 Values for E_r=10

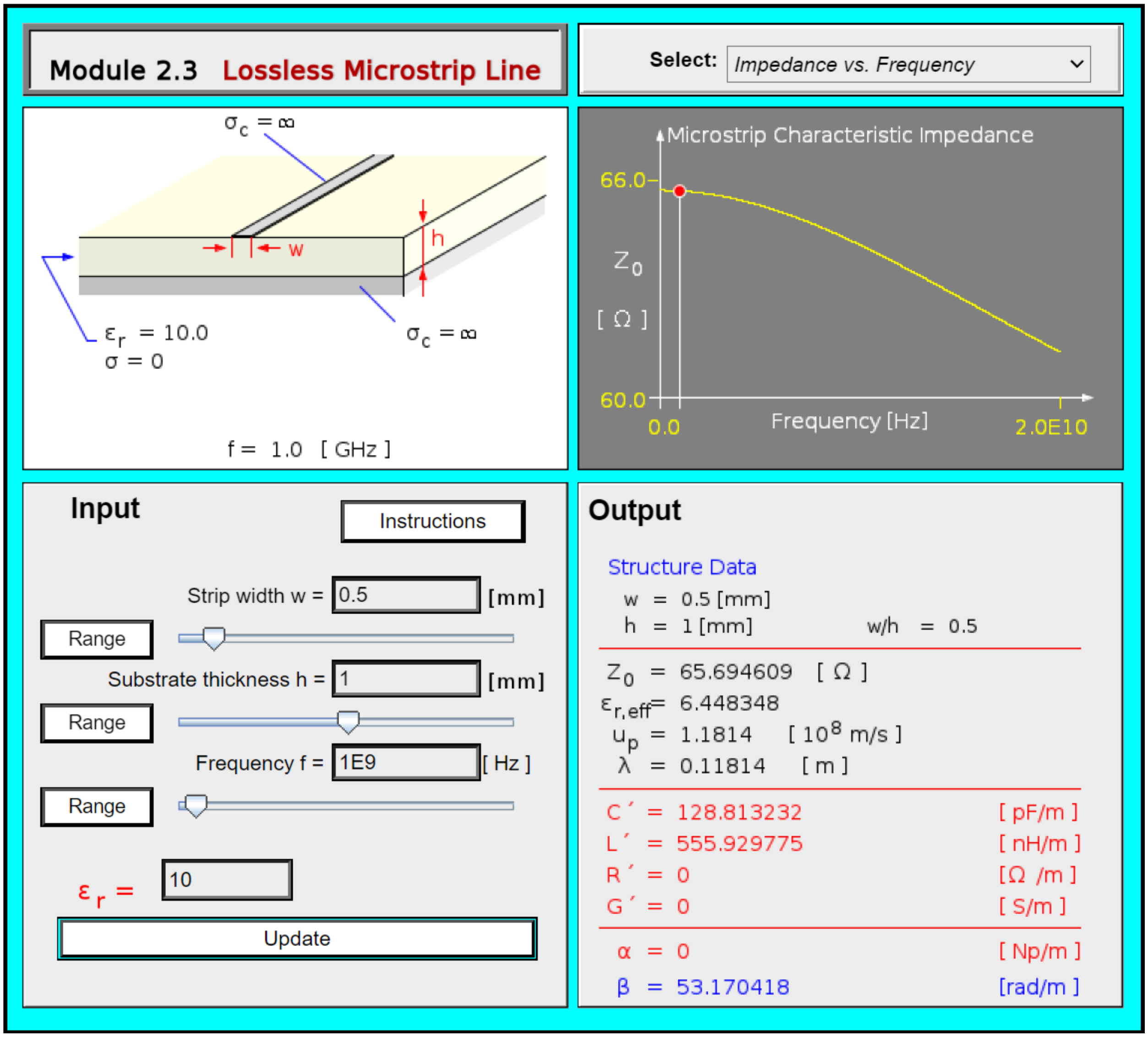

Figure 2.0 - Screenshot of Module 2.3 App for S=0.5

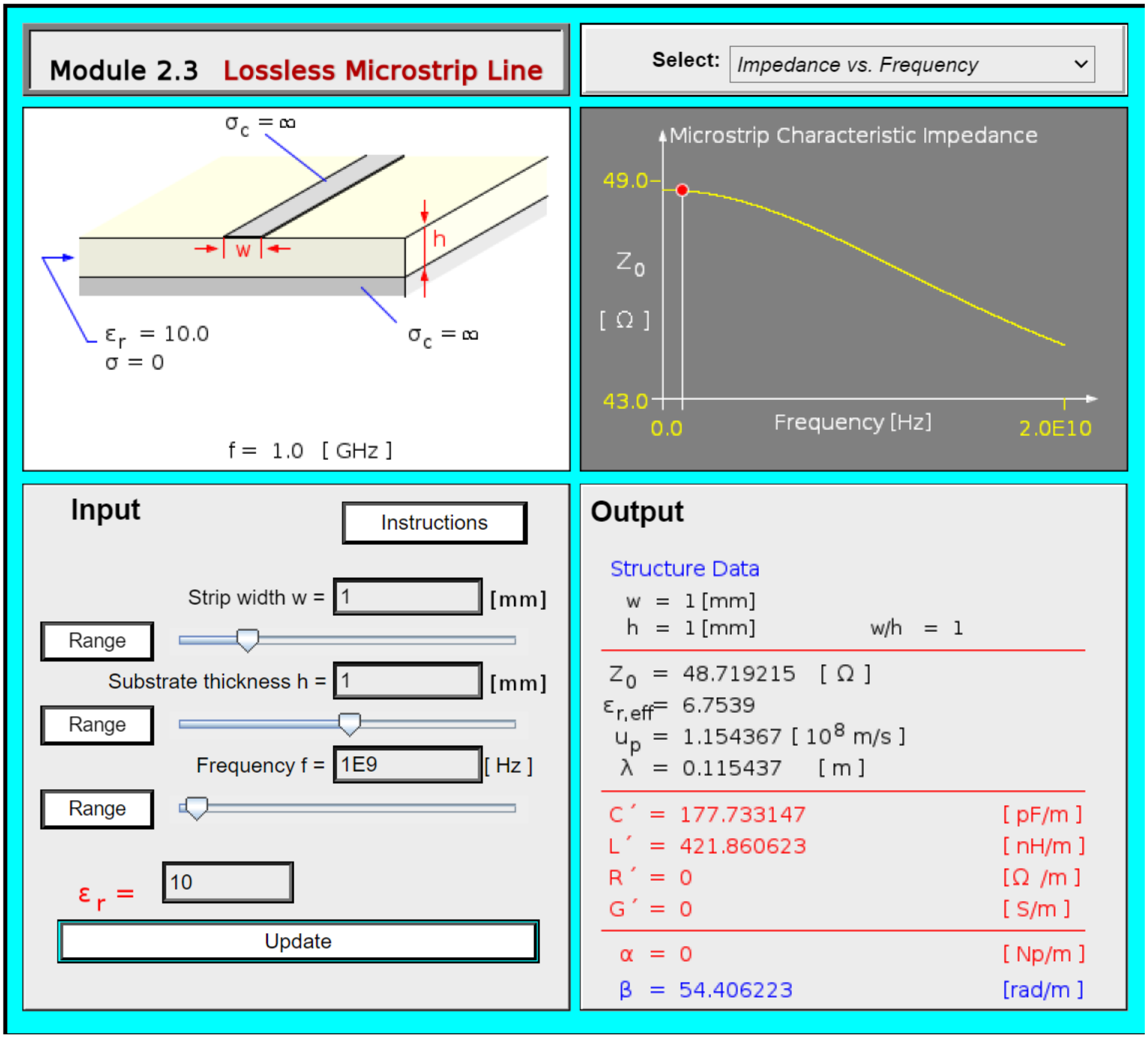

Figure 2.1 - Screenshot of Module 2.3 App for S=1.0

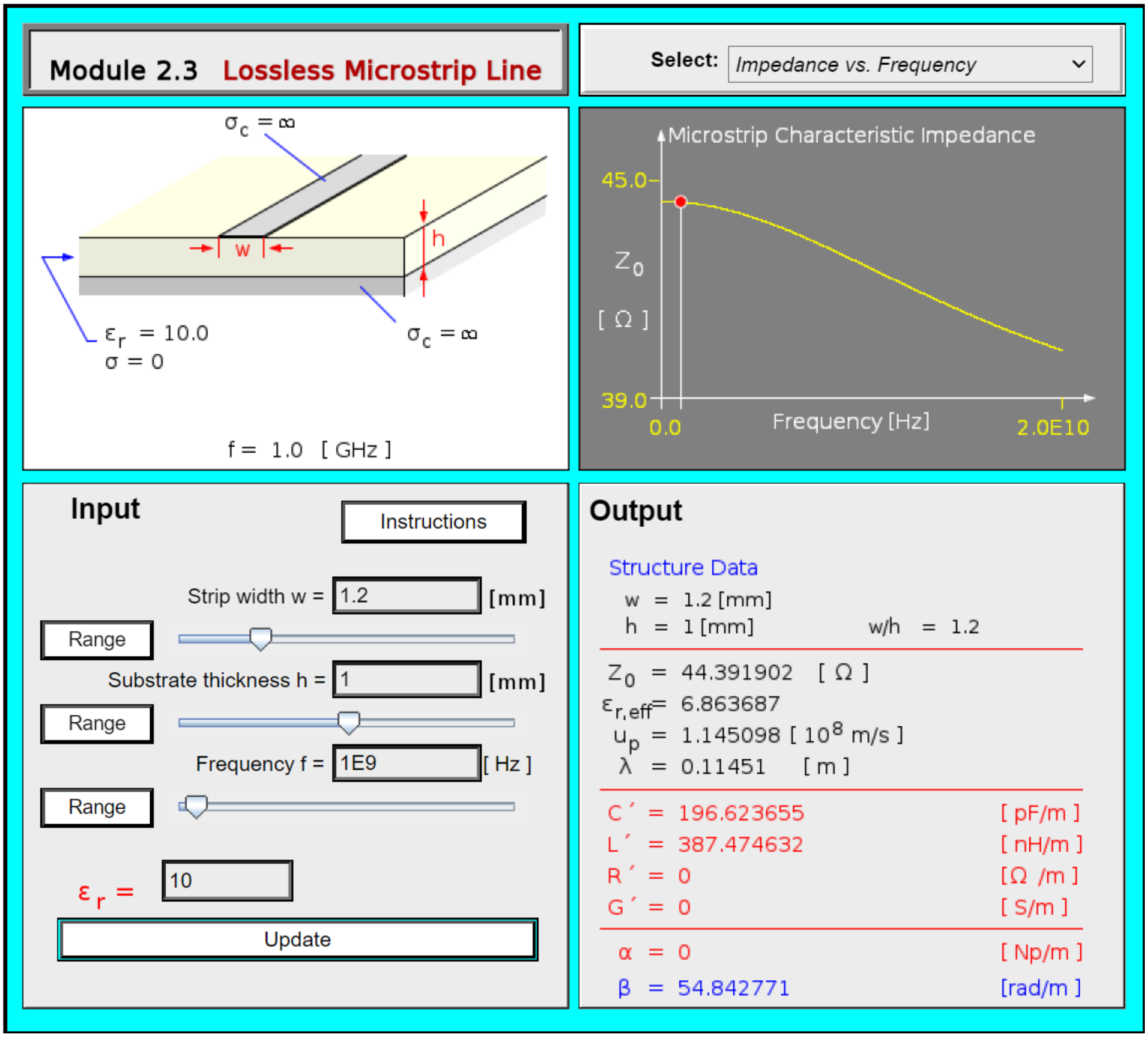

Figure 2.2 - Screenshot of Module 2.3 App for S=1.2

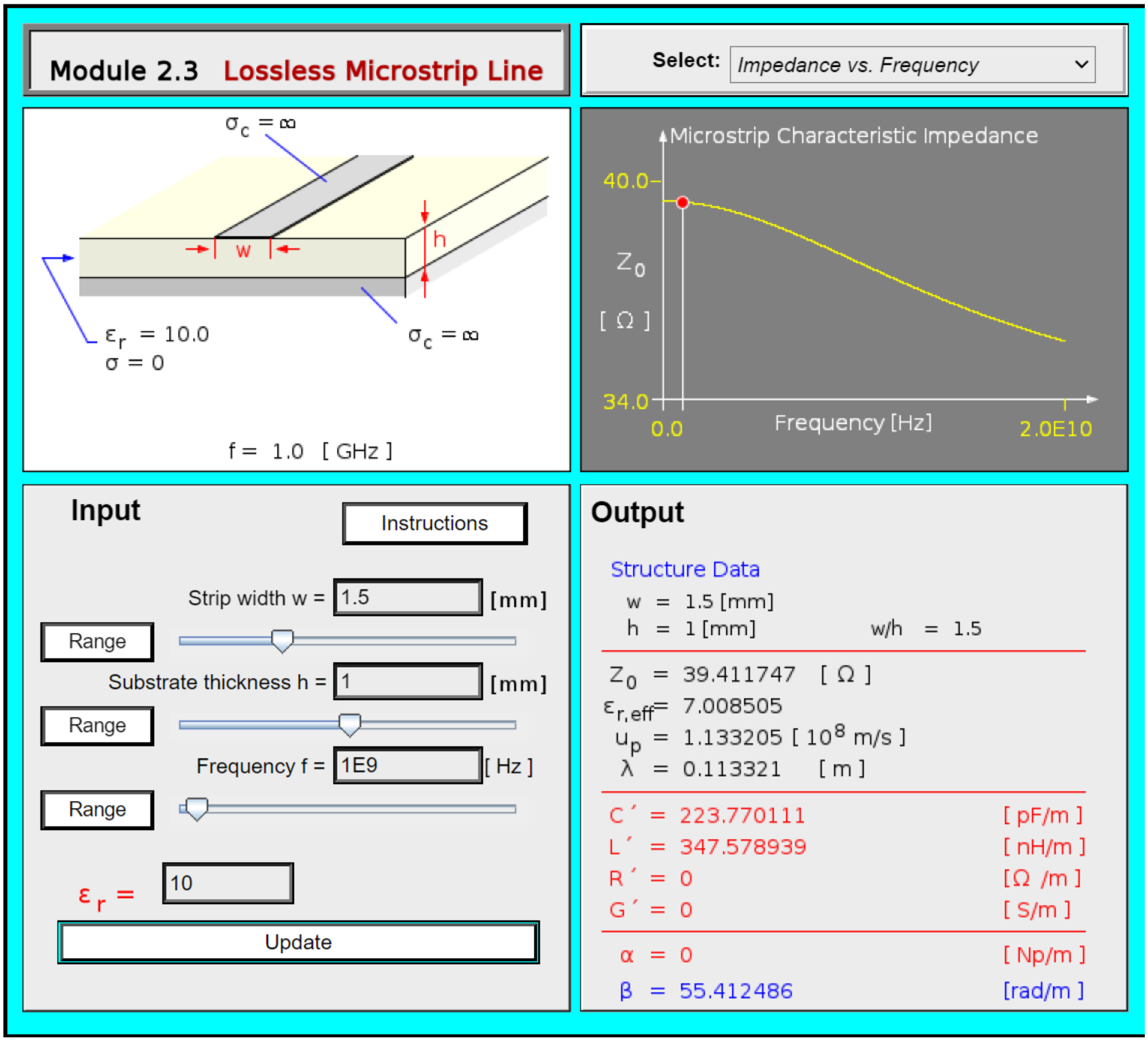

Figure 2.3 - Screenshot of Module 2.3 App for S=1.5

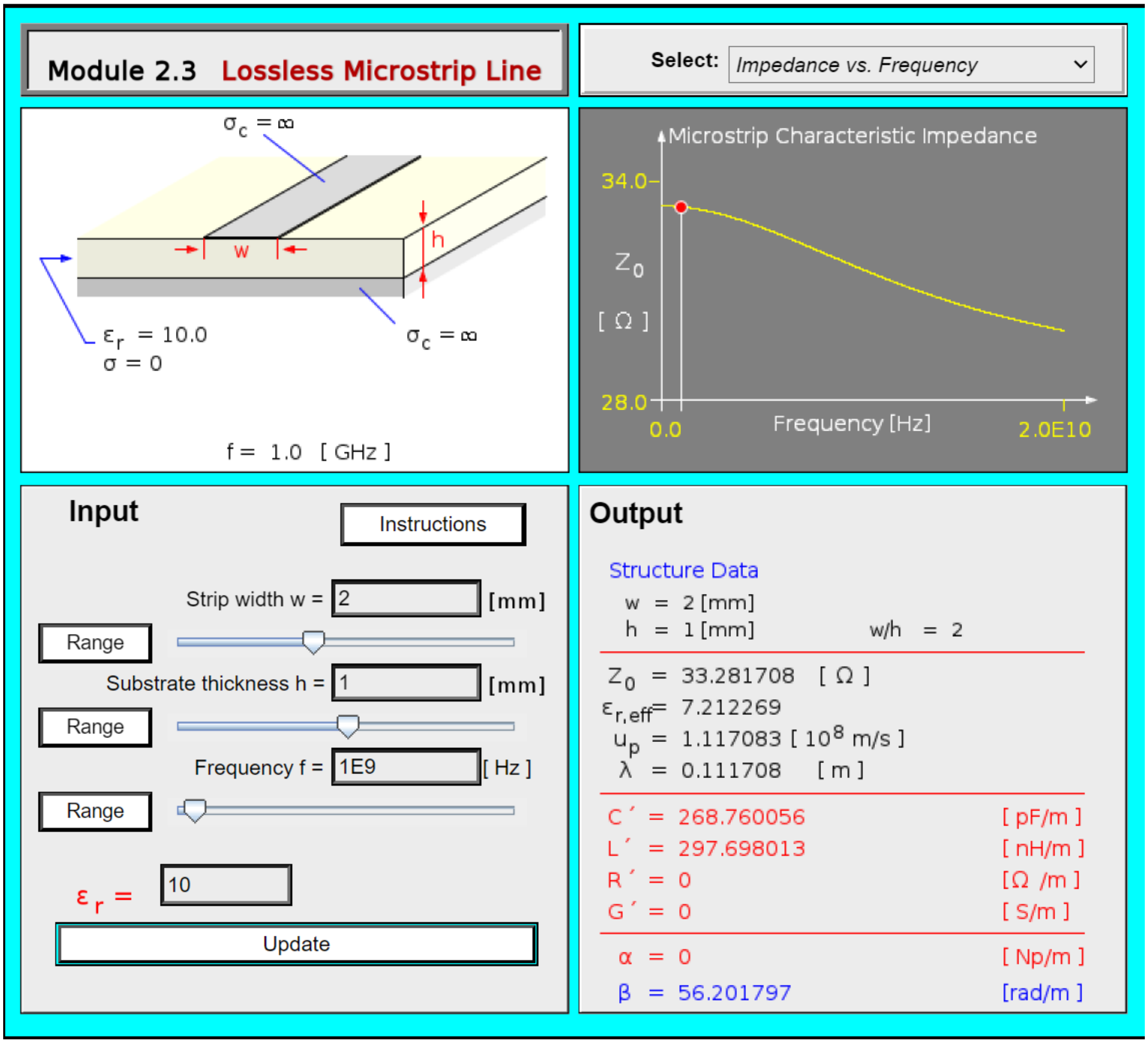

Figure 2.4 - Screenshot of Module 2.3 App for S=2.0

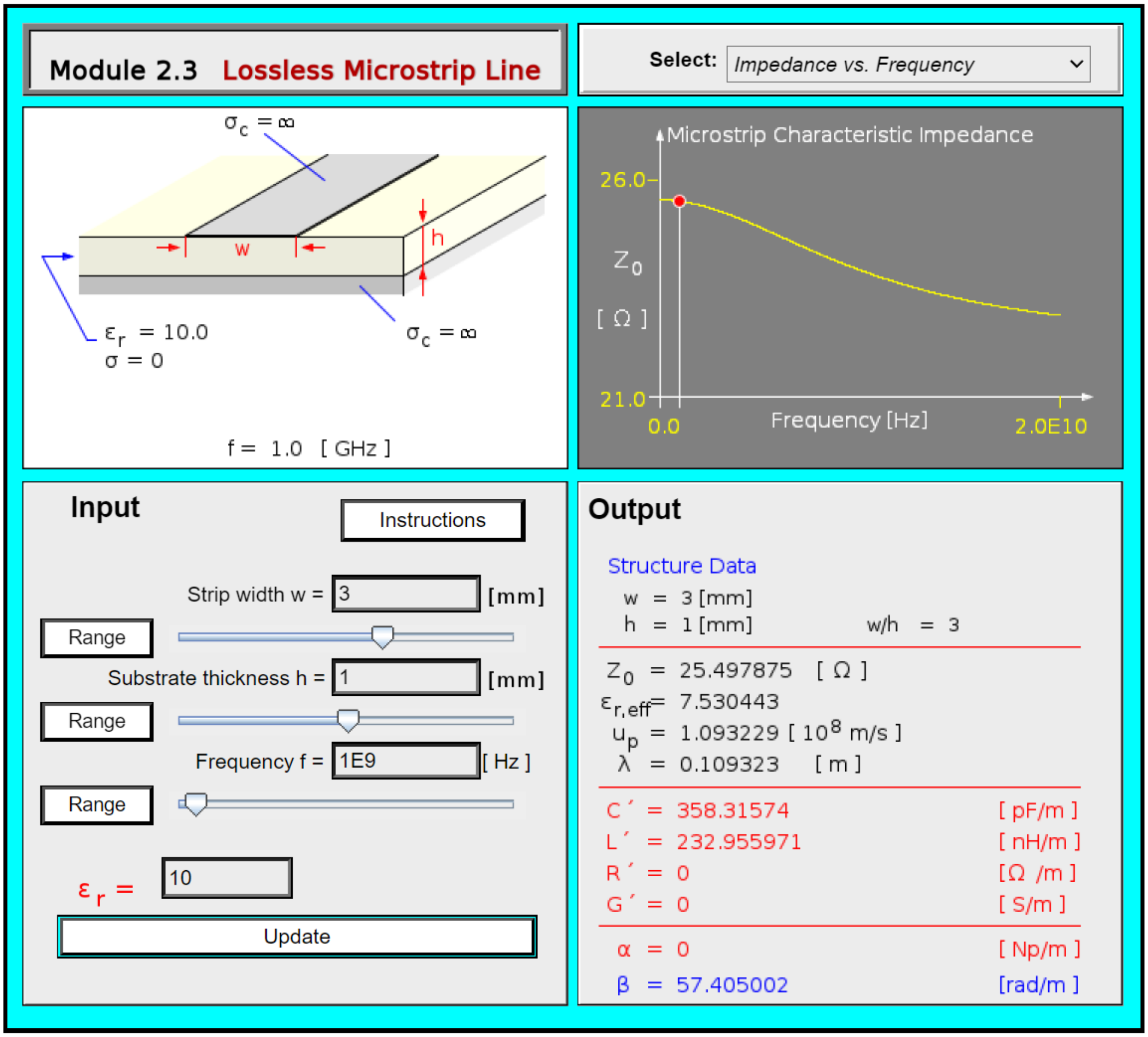

Figure 2.5 - Screenshot of Module 2.3 App for S=3

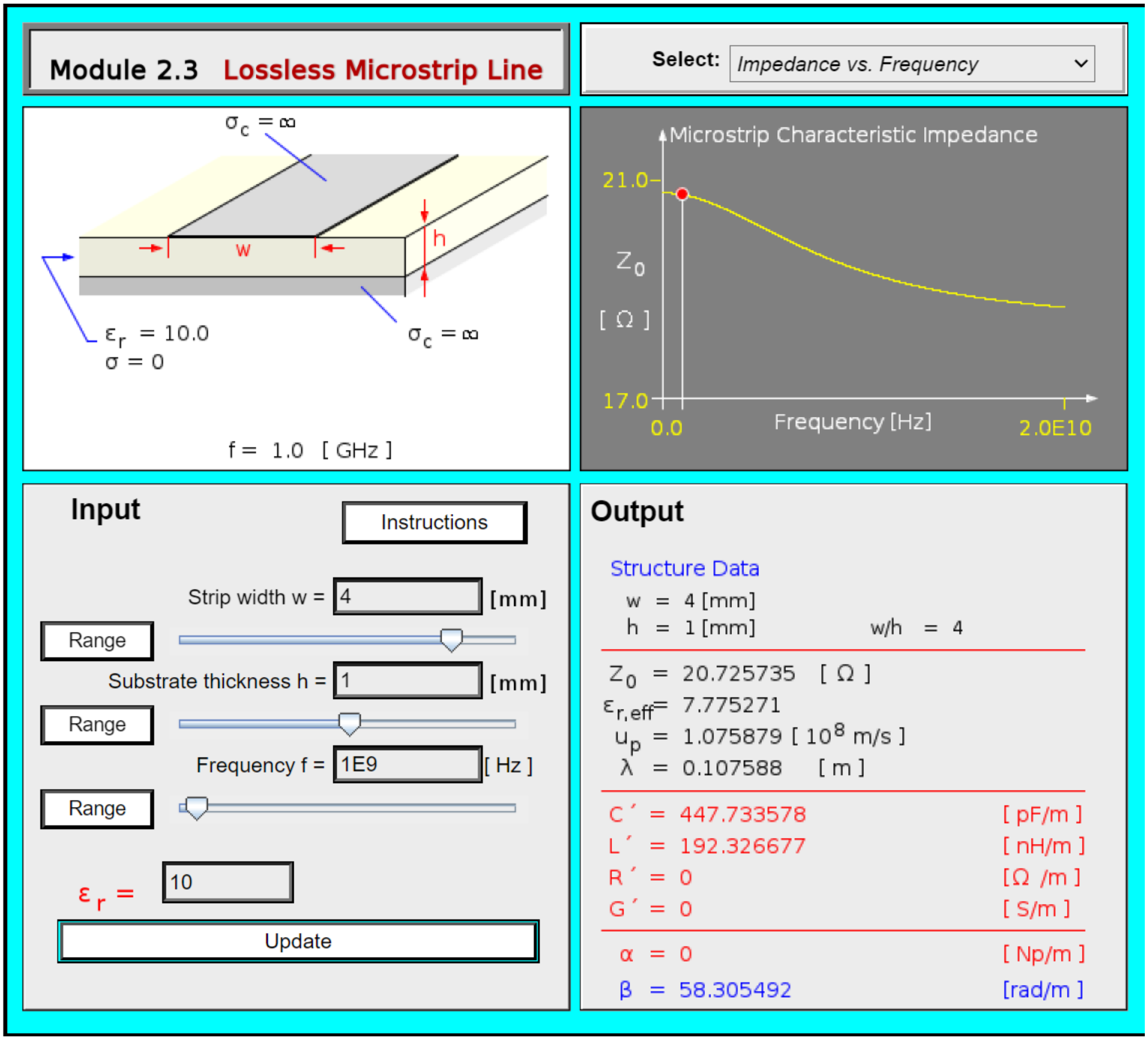

Figure 2.6 - Screenshot of Module 2.3 App for S=4

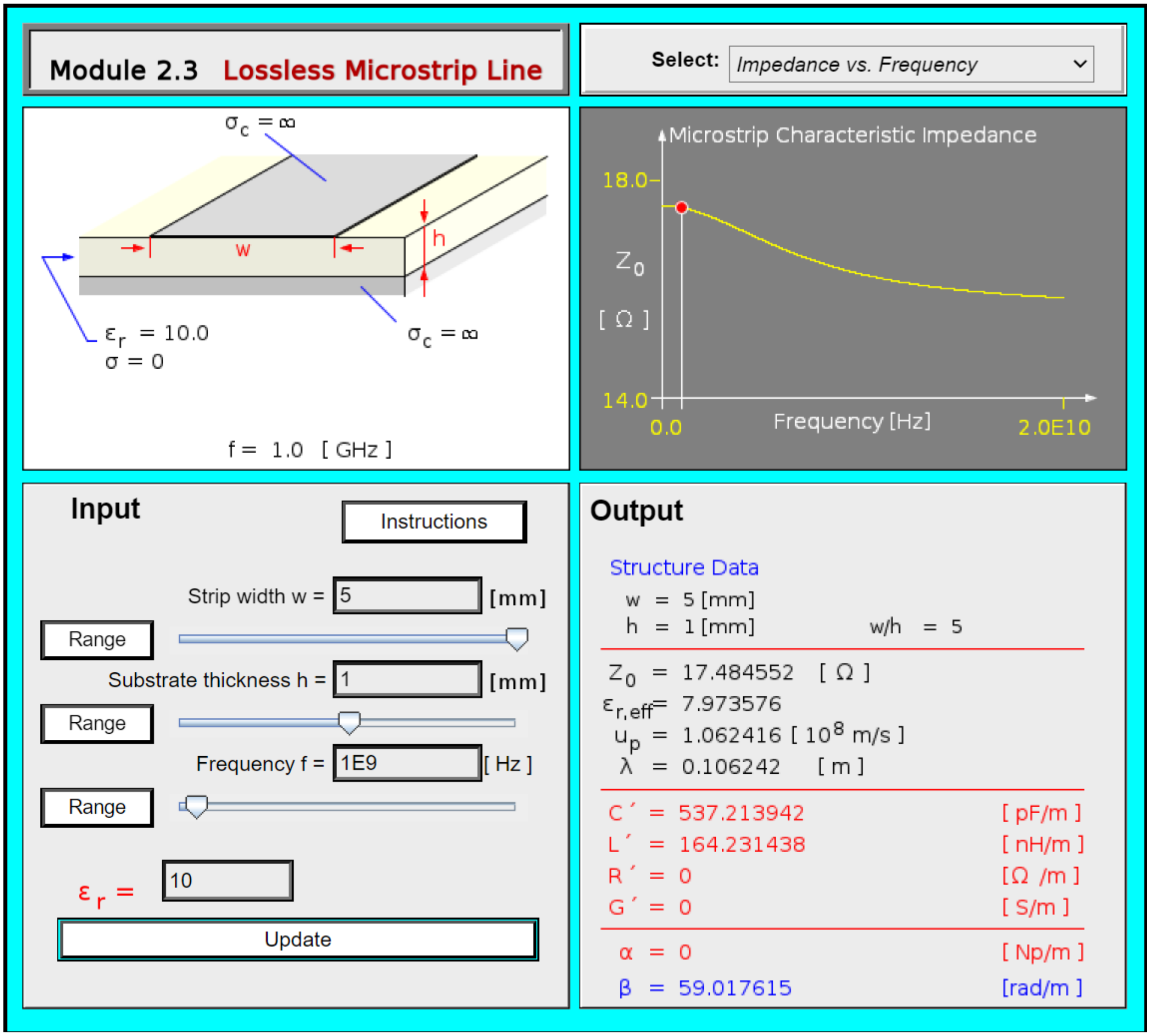

Figure 2.7 - Screenshot of Module 2.3 App for S=5

Code Appendix

The following code was used to create Table 1.0 and Figure 1.0, Figure 1.1 and Figure 1.2.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

c = 3e8

def calculate_x(Er):
    return 0.56 * ((Er - 0.9) / (Er + 3))**0.05

def calculate_y(s):
    return 1 + 0.02 * np.log((s**4 + 0.00037 * s**2) / (s**4 + 0.43)) +
0.05 * np.log(1 + 0.00017 * s**3)

def calculate_Eeff(Er, s, x, y):
    return (Er + 1) / 2 + (Er - 1) / 2 * (1 + (10) / s)**(-x * y)

def calculate_Z0(Eeff, s):
    t = (30.67 / s)**0.75
    return 60 / np.sqrt(Eeff) * np.log((6 + (2 * np.pi - 6) * np.exp(-t))
/ s + np.sqrt(1 + 4 / s**2))

def approximation_a(Z0, Er):
    q = (60 * np.pi**2) / (Z0 * np.sqrt(Er))
    return (2 / np.pi) * ((q - 1) - np.log(2 * q - 1) + (Er - 1) / (2 *
Er) * (np.log(q - 1) + 0.29 - 0.52 / Er))

def approximation_b(Z0, Er):
    p = np.sqrt((Er + 1) / 2) * (Z0 / 60) + (Er - 1) / (Er + 1) * (0.23 +
0.12 / Er)
    return (8 * np.exp(p)) / (np.exp(2 * p) - 2)

Er_10 = 10
h = 1e-3
s_values_10 = np.array([0.5, 1.0, 1.2, 1.5, 2.0, 3.0, 4.0, 5.0])
provided_Z0_values_10 = np.array([65.711672, 48.740235, 44.414188,
39.435683, 33.30811, 25.528372, 20.759498, 17.520966])
```

```python
Er_2 = 2
s_values_2 = np.arange(0.5, 5.1, 0.1)
Z0_values_2 = []

for s in s_values_2:
    x = calculate_x(Er_2)
    y = calculate_y(s)
    Eeff = calculate_Eeff(Er_2, s, x, y)
    Z0_values_2.append(calculate_Z0(Eeff, s))

reverse_engineered_s_values_a_10 = [approximation_a(Z0, Er_10) for Z0 in
provided_Z0_values_10]
reverse_engineered_s_values_b_10 = [approximation_b(Z0, Er_10) for Z0 in
provided_Z0_values_10]
reverse_engineered_s_values_a_2 = [approximation_a(Z0, Er_2) for Z0 in
Z0_values_2]
reverse_engineered_s_values_b_2 = [approximation_b(Z0, Er_2) for Z0 in
Z0_values_2]

relative_differences_a_10 = 100 *
(np.array(reverse_engineered_s_values_a_10) - s_values_10) / s_values_10
relative_differences_b_10 = 100 *
(np.array(reverse_engineered_s_values_b_10) - s_values_10) / s_values_10

plt.figure(figsize=(12, 6))
plt.plot(provided_Z0_values_10, relative_differences_a_10, '-o',
label='Approximation A')
plt.plot(provided_Z0_values_10, relative_differences_b_10, '-o',
label='Approximation B')
plt.xlabel('Characteristic Impedance Z0 (Ohms)')
plt.ylabel('Relative Difference in s (%)')
plt.title('Relative Difference in s for Er=10')
plt.legend()
plt.grid(True)
plt.savefig('Relative_Difference_Er10.png')
plt.show(block=False)

relative_differences_a_2 = 100 *
(np.array(reverse_engineered_s_values_a_2) - s_values_2) / s_values_2
```

```
relative_differences_b_2 = 100 *
(np.array(reverse_engineered_s_values_b_2) - s_values_2) / s_values_2

plt.figure(figsize=(12, 6))
plt.plot(Z0_values_2, relative_differences_a_2, '-o', label='Approximation
A')
plt.plot(Z0_values_2, relative_differences_b_2, '-o', label='Approximation
B')
plt.xlabel('Characteristic Impedance Z0 (Ohms)')
plt.ylabel('Relative Difference in s (%)')
plt.title('Relative Difference in s for Er=2')
plt.legend()
plt.grid(True)
plt.show(block=False)

calculated_Z0_values_10 = [calculate_Z0(calculate_Eeff(Er_10, s,
calculate_x(Er_10), calculate_y(s)), s) for s in s_values_10]
percent_diff_Z0_10 = 100 * (calculated_Z0_values_10 -
provided_Z0_values_10) / provided_Z0_values_10

plt.figure(figsize=(12, 6))
plt.plot(s_values_10, percent_diff_Z0_10, '-o')
plt.xlabel('Width-to-Thickness Ratio s')
plt.ylabel('Percent Difference in Z0 (%)')
plt.title('Percent Difference Between Simulated and Calculated Z0 Values
for Er=10')
plt.grid(True)
plt.savefig('Percent_Difference_Z0_Er10.png')
plt.show(block=False)

Z0_values_2 = np.array(Z0_values_2)
valid_range_a_10 = provided_Z0_values_10[np.abs(relative_differences_a_10)
<= 2]
valid_range_b_10 = provided_Z0_values_10[np.abs(relative_differences_b_10)
<= 2]
valid_range_a_2 = Z0_values_2[np.abs(relative_differences_a_2) <= 2]
valid_range_b_2 = Z0_values_2[np.abs(relative_differences_b_2) <= 2]

print("Valid range for Approximation A for Er=10:", valid_range_a_10)
print("Valid range for Approximation B for Er=10:", valid_range_b_10)
```

```python
print("Valid range for Approximation A for Er=2:", valid_range_a_2)
print("Valid range for Approximation B for Er=2:", valid_range_b_2)

data = {
    'Valid Range A for Er=10': pd.Series(valid_range_a_10),
    'Valid Range B for Er=10': pd.Series(valid_range_b_10),
    'Valid Range A for Er=2': pd.Series(valid_range_a_2),
    'Valid Range B for Er=2': pd.Series(valid_range_b_2)
}

df = pd.DataFrame(data)
excel_filename = "results_a3.xlsx"
df.to_excel(excel_filename, index=False, engine='openpyxl')
print(f"Data saved to {excel_filename}")
plt.show()
```