Università degli studi di Padova

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE
Corso di laurea in Ingegneria dell'Informazione

Elaborato finale

Solutore numerico per le linee di trasmissione

Candidato:
Luca Bonaventura
Matricola 1216534

Relatore:
Prof. Marco Santagiustina

Indice

Elenco delle figure

Capitolo 1

Introduzione

Il modello di linea di trasmissione è fondamentale per la progettazione di sistemi di comunicazione. Esso è infatti usato per sfruttare al meglio una linea, permettendo di minimizzare fenomeni di riflessione dell'onda elettromagnetica all'interno della linea. Lo scopo della tesi è quello di illustrare il funzionamento del simulatore sviluppato per analizzare questo tipo di dinamiche. In dettaglio lo scopo del programma è quello di risolvere linee di trasmissione, per trovare in modo rapido e preciso come si propaga l'onda elettromagnetica all'interno della linea di trasmissione, oltre a trovare parametri di dimensionamento ideale di vari adattatori di impedenza. Il modello usato è quello di linea senza perdite, ed il programma è sviluppato tramite un applicativo Matlab. Verrà qui data una breve descrizione del modello usato e poi delle caratteristiche del programma.

1.1 Linee di trasmissione

In generale si modellizza una linea bifilare nel seguente modo (figura 1.1):

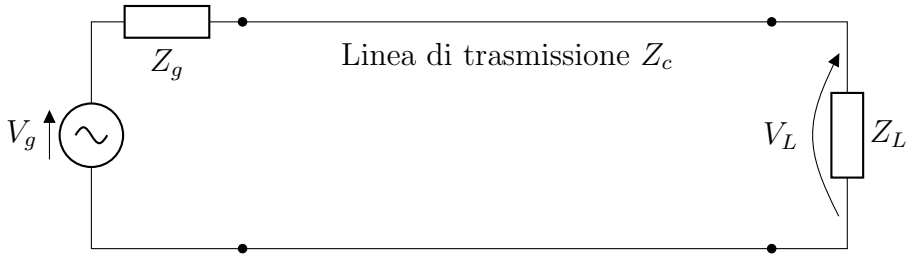

Figura 1.1: Schema di una linea di trasmissione

Il programma considera una linea senza perdite, con costante di fase $\gamma = \jmath\beta$ con $\beta \in \Re$ ed impedenza caratteristica $Z_c \in \Re$. Su questa ipotesi la soluzione delle

equazioni d'onda di tensione e corrente è:

$$V(-l) = V_+ e^{j\beta l} + V_- e^{-j\beta l} = V_+(e^{j\beta l} + \Gamma e^{-j\beta l}) \tag{1.1}$$

$$I(-l) = \frac{V_+}{Z_c} e^{j\beta l} - \frac{V_-}{Z_c} e^{-j\beta l} = \frac{V_+}{Z_c}(e^{j\beta l} - \Gamma e^{-j\beta l}) \tag{1.2}$$

dove abbiamo definito il coefficiente di riflessione $\Gamma = \frac{V_-}{V_+} = \frac{Z_L - Z_c}{Z_L - Z_c}$ e $-l$ è la distanza del punto della linea dal carico.
Per l'impedenza d'ingresso Z_{in}, definita come rapporto tra la tensione d'ingresso e la corrente d'ingresso, si trova:

$$Z_{in}(-l) = \frac{V_{in}}{I_{in}} = Z_c \frac{1 + \Gamma e^{-2j\beta l}}{1 - \Gamma e^{-2j\beta l}} \tag{1.3}$$

Tramite semplici passaggi algebrici si trova:

$$Z(-l) = Z_c \frac{Z_L + \jmath Z_c \tan(\beta l)}{Z_c + \jmath Z_L \tan(\beta l)} \tag{1.4}$$

Si partirà da questi risultati per la dimostrazione delle relazioni sugli adattatori del capitolo 4.

1.2 Obiettivi del programma

Specificati i parametri di una linea, il programma è in grado di risolvere problemi relativi alle linee di trasmissione. In particolare l'applicativo può mostrare l'andamento nello spazio di parametri come la tensione in modulo e fase (ved. cap. 2.2 per tutte le funzionalità), oltre a calcolare parametri come la lunghezza d'onda, il coefficiente di riflessione, ecc. L'applicazione è inoltre utile per dimensionare alcuni tra i sistemi di adattamento più diffusi, quali l'adattatore a quarto d'onda e lo STUB.

Capitolo 2

Operazioni preliminari per l'utilizzo

2.1 Installazione dell'applicazione

Il modo più veloce per poter usare l'applicazione è copiare il codice su Matlab (il codice è riportato nell'appendice A). Questa strategia necessita naturalmente di avere a disposizione Matlab, programma non molto diffuso fuori dall'ambito accademico per il costo delle licenze. Per ovviare a questo problema è stata prodotta una versione "stand alone" dell'applicazione. Con questa strategia è sufficiente cliccare sull'eseguibile e seguire l'installazione guidata per poter installare il programma. Per poter ricevere il file eseguibile è sufficiente scrivere una mail a luca.bonaventura(at)studenti.unipd.it .

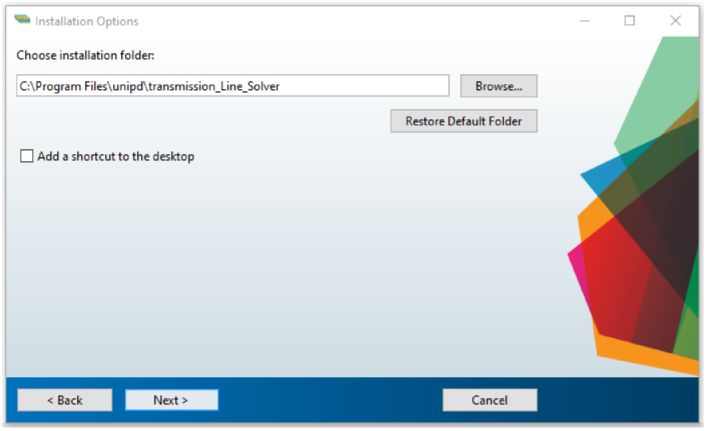

Figura 2.1: Installazione guidata dell'applicazione

2.2 L'interfaccia grafica

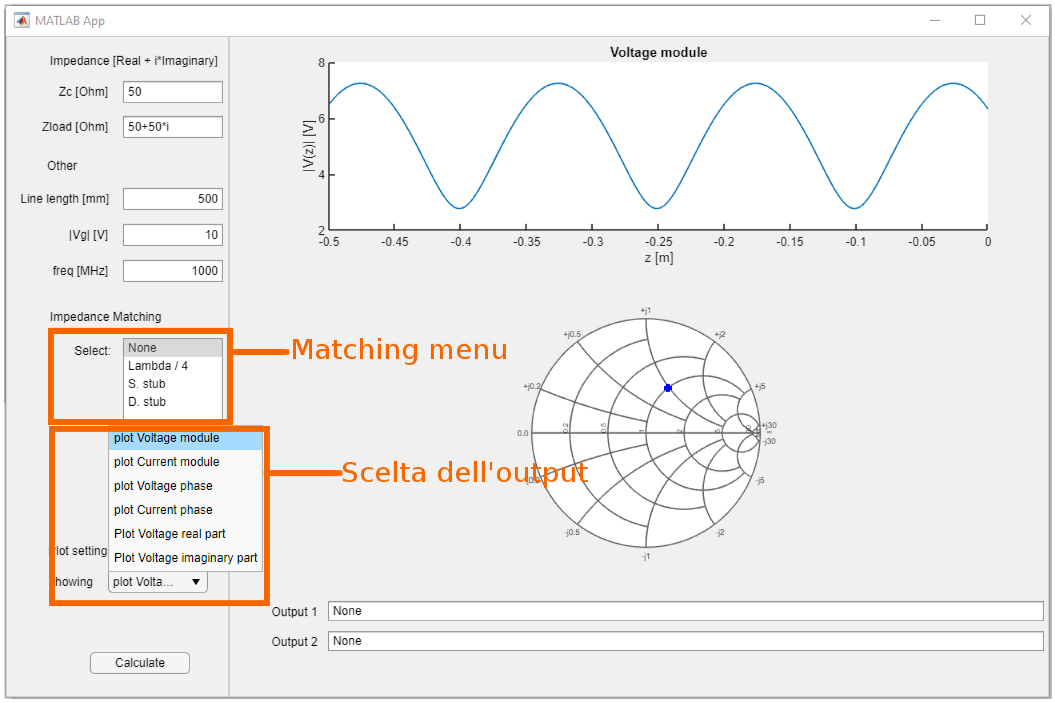

Figura 2.2: Funzionalità principali dell'applicazione

Aprendo il programma viene mostrata la finestra di figura (2.2), che si suddivide in due parti: la sezione di sinistra è dedicata all'inserimento dei parametri della linea e di ciò che si vuole visualizzare, mentre quella di destra all'output.
Nella parte di sinistra è possibile inserire:

- Zc → per settare l'impedenza caratteristica della linea;
- Zload → per settare l'impedenza del carico della linea;
- Line length → per indicare la lunghezza della linea;
- Vg→ per settare il modulo della tensione del generatore collegato alla linea;
- freq → per settare la frequenza dell'onda trasmessa nella linea;
- Impedance Matching → per scegliere il tipo di adattatore d'impedenza da inserire nella linea;
- Plot settings → per scegliere quale grandezza visualizzare nel grafico.

Sulla destra è invece possibile visualizzare:

- Il grafico della grandezza scelta per la visualizzazione. Si noti che l'asse delle ascisse rappresenta la posizione nella linea. In $z = 0$ è collegato il carico, mentre in $z = -l$ è collegato il generatore;
- La carta di Smith con la posizione dell'impedenza normalizzata del carico;
- Due campi con l'output di tipo testuale, dove vengono mostrati:
  - La lunghezza d'onda dell'onda trasmessa nella linea, solo quando viene cambiato il valore della frequenza;
  - Il rapporto d'onda stazionaria SWR, solo quando viene cambiato il valore dell'impedenza caratteristica o di quella del carico;
  - Il tipo di adattamento scelto, solo quando viene cambiata la selezione;
  - I parametri dei dimensionamenti dell'adattatore scelto. Gli adattatori a singolo e a doppio STUB hanno ciascuno due soluzioni distinte per adattare la linea, quindi vengono visualizzate entrambe. Il primo viene stampato su Output 1 mentre il secondo su Output 2.

Capitolo 3

L'implementazione del software

Vengono qui presentate le formule usate per produrre l'output.

3.1 Calcolo di tensioni e correnti

3.1.1 I grafici

In questa sezione verranno presentati i parametri che si possono visualizzare nel grafico. Le funzioni usate in questo capitolo derivano direttamente dalle equazioni del telegrafo 1.1 e 1.2 presentate nel paragrafo 1.1.

- Selezionando "plot Voltage module" è possibile visualizzare il modulo della tensione $|V(z)|$. Il codice Matlab della funzione è il seguente:

```
V=abs(V0p.*(exp(-1j*beta.*z)+(gamma*exp(1j*beta.*z))))
```

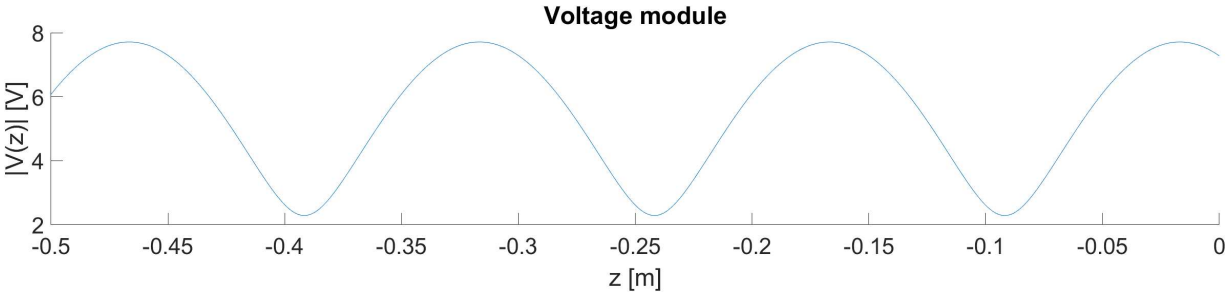

Figura 3.1: Modulo della tensione lungo la linea con $Z_c = 50\Omega$ e $Z_L = 75 + \jmath 75\Omega$

- Selezionando "plot Current module" è possibile visualizzare il modulo della corrente $|I(z)|$. Il codice Matlab della funzione è il seguente:

```
I=abs(V0p/Zc.*(exp(-j*beta.*z)-(gamma*exp(j*beta.*z))
```

- Selezionando "plot Voltage phase" è possibile visualizzare la fase della tensione $\angle V(z)$. Il codice Matlab della funzione è il seguente:

```
V=angle(V0p.*(exp(-j*beta.*z)+(gamma*exp(j*beta.*z))))
```

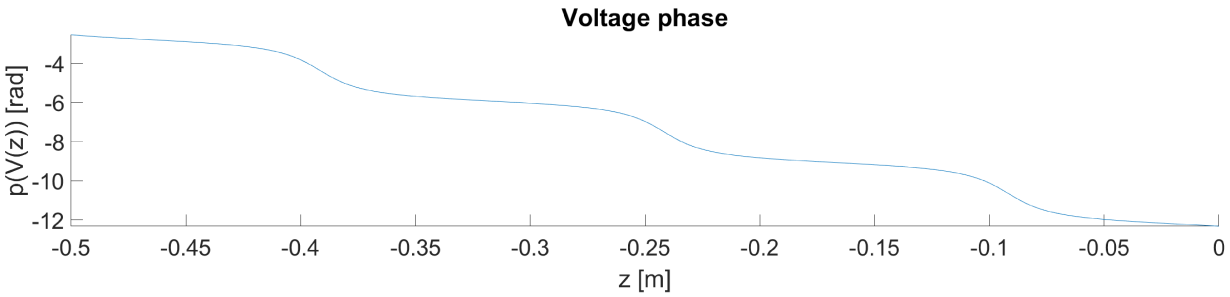

Figura 3.2: Fase della tensione lungo la linea con $Z_c = 50\Omega$ e $Z_L = 75 + \jmath 75\Omega$

- Selezionando "plot Current phase" è possibile visualizzare la fase della corrente $\angle I(z)$. Il codice Matlab della funzione è il seguente:

```
I=angle(V0p/Zc.*(exp(-j*beta.*z)-(gamma*exp(j*beta.*z)
```

3.1.2 Parametri utili mostrati

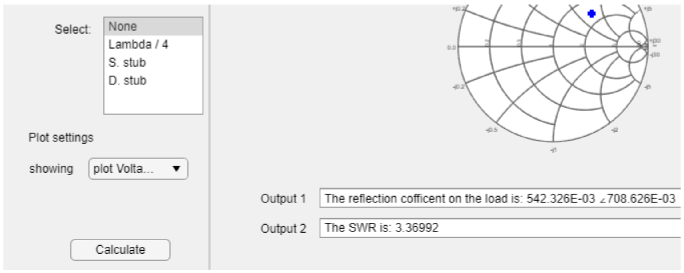

Figura 3.3: Coefficiente di riflessione e SWR mostrati lungo la linea con $Z_c = 50\Omega$ e $Z_L = 75 + \jmath 75\Omega$

Come detto in precedenza, ogni volta che vengono modificati dei parametri della linea vengono mostrati in output alcuni parametri, che vengono qui descritti in dettaglio:

- Quando viene cambiato il valore della frequenza viene automaticamente calcolata la lunghezza d'onda dell'onda trasmessa nella linea, secondo la relazione:

$$\lambda = \frac{u_p}{f} = \frac{c}{f}\frac{1}{\sqrt{\varepsilon_r}} \tag{3.1}$$

dove solitamente la permettività relativa $\varepsilon_r = 1$. La funzione Matlab usata è:

```
lambda=physconst('LightSpeed')/f*1/sqrt(epsilonr)
```

Si noti che il valore di ε_r è impostato di default a 1 ma può essere modificato all'interno del codice tra le costanti del programma.

- Il coefficiente di riflessione, mostrato solo quando viene cambiato il valore o dell'impedenza caratteristica o di quella del carico, viene calcolato tramite:

$$\Gamma = \frac{V_-}{V_+} = \frac{Z_L - Z_c}{Z_L - Z_c} \tag{3.2}$$

La funzione Matlab usata è:

```
gamma = z2gamma(Zl,Zc)
```

Assieme al coefficiente di riflessione viene anche mostrato il rapporto d'onda stazionaria (SWR):

$$S = \frac{|V_{max}|}{|V_{min}|} = \frac{1 + |\Gamma|}{1 - |\Gamma|} \tag{3.3}$$

- Il tipo di adattamento scelto, solo quando viene cambiata la selezione;
- I parametri dei dimensionamenti dell'adattatore scelto. Gli adattatori a singolo e a doppio STUB hanno ciascuno due soluzioni distinte per adattare la linea, quindi vengono visualizzate entrambe. Il primo viene stampato su Output 1 mentre il secondo su Output 2 (ved. 3.2).

3.2 Progettazione di adattatori di impedenza

Viene qui descritto come il software calcola il dimensionamento degli adattamenti, la dimostrazione dei risultati usati è riportata nel capitolo successivo.

3.2.1 Adattatore a quarto di lunghezza d'onda

In questo caso per adattare la linea usiamo un tratto di linea di una precisa impedenza Z_c' posto a distanza l' dal carico. L'impedenza del tratto di linea dovrà essere:

$$Z_c' = \frac{Z_c^2}{Z_{in}} = Z_c\sqrt{S} \tag{3.4}$$

Mentre la distanza dal carico in cui porlo sarà:

$$l_1' = -\frac{\angle\Gamma}{2\beta} + a\frac{\pi}{2\beta} \tag{3.5}$$

con $a \in \mathbb{N}$ e a t.c. $l_1' \geq 0$. Il codice della funzione è:

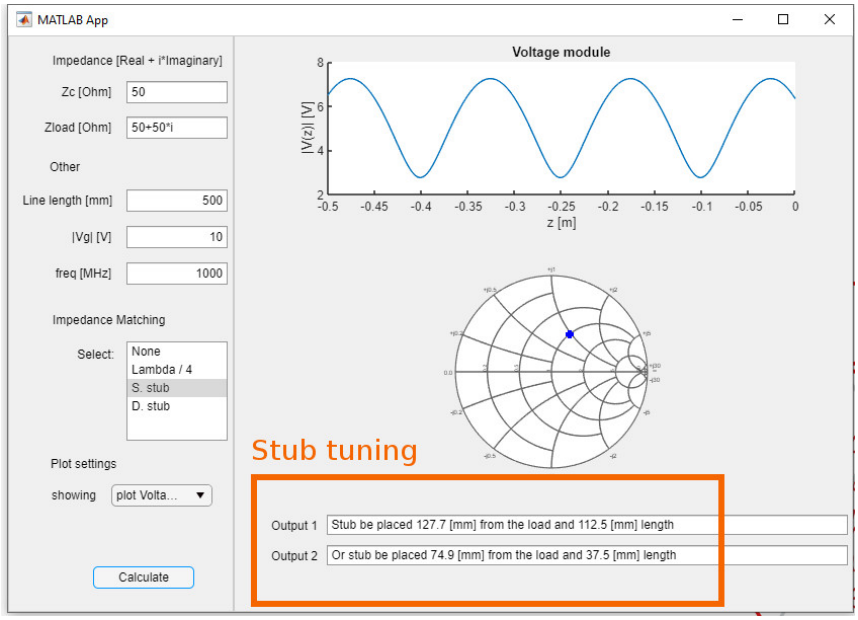

Figura 3.4: Esempio di dimensionamento dell'adattatore a singolo stub

```
dist1=angle(gamma)/(2*beta); %In meter, dist from the
   first maximum
if(dist1<0)
        dist1=dist1+lambda/2;
end
S=(1+abs(gamma))/(1-abs(gamma)); %ROS
Zc21=Zc*sqrt(S);      %In the maximum of voltage
```

3.2.2 Adattatore a singolo stub

Nell'adattatore a singolo STUB aggiungiamo un tratto di linea di impedenza Z_c di una precisa lunghezza l', collegato in parallelo a distanza d dal carico. La relazione per trovare questi valori è:

$$l' = \frac{1}{\beta} \cdot \arctan\left(-\frac{1}{\mathfrak{Im}(y(-d))} \right) + N \cdot \frac{\lambda}{2} \tag{3.6}$$

$$d = \frac{1}{\beta} \cdot \arctan \frac{2b \pm \sqrt{4b^2 - 4(b^2 + a^2 - a) \cdot (1 - a)}}{2 \cdot (b^2 + a^2 - a)} + N \cdot \frac{\lambda}{2} \tag{3.7}$$

dove $N \in \mathbb{N}$ e N è t.c. $d \geq 0$, mentre $a = \mathfrak{Re}(y(-d))$ e $b = \mathfrak{Im}(y(-d))$.
La funzione é:

```
a=real(Zc/Zl);
b=imag(Zc/Zl);
```

```
yl=Zc/Zl;     %load ammetance
d1=0;
if(real(Zc/Zl) ~= 1 && real(Zl/Zc) ~= 1)
        d1=1/beta*atan((2*b+sqrt(4*b^2-4*(b^2+a^2-a)*(1-a)
           ) )/(2*(b^2+a^2-a)));
end
if(real(Zc/Zl) ~= 1 && real(Zl/Zc) == 1)
     d1=lambda/4+lambda*0.176;
end
if(d1<0)
    d1=d1+lambda/2;
end
%distance from the first point with RE{Y(l11)}=RE{Yc} [m]
b1=imag((yl+1i*tan(beta*d1))/(1+1i*tan(beta*d1)*yl));
bstub1=-b1;  %ammetance that the STUB must create
l1=1/beta*atan(-1/bstub1);
if(l1<0)
        l1=l1+lambda/2;
end
```

3.2.3 Adattatore a doppio STUB

In quest'ultimo adattatore sfruttiamo due linee di impedenza Z_c collegate in parallelo a distanza dal carico fissata a priori. Le distanze a cui porre gli STUB possono essere modificate a piacimento all'interno del codice, si noti che d_1 è la distanza tra carico e primo STUB, mentre d_2 è la distanza tra primo e secondo STUB. Si noti che solitamente nei casi pratici $d_1 = 0$ e $d_2 = \lambda/8, \lambda/4, 3\lambda/8$, tuttavia le funzioni sono state sviluppate per d_1 e d_2 generici, in modo da essere più generali. Di default l'applicazione ha $d_1 = 0$ e $d_2 = \lambda/8$, tuttavia queste due grandezze possono essere modificate a proprio piacimento agendo nel codice dell'applicazione. Le lunghezze delle linee sono date dalle seguenti relazioni:

$$l_2 = \frac{1}{\beta} \cdot \arctan\left(-\frac{1}{b - b_2} \right) \tag{3.8}$$

$$l_1 = \frac{1}{\beta} \cdot \arctan\left(-\frac{1}{-b_1} \right) \tag{3.9}$$

dove:

$$b_2 = \frac{a_1 \pm \sqrt{a_1\big(1+\tan^2(-\beta d_2)\big) - a_1^2\tan^2(-\beta d_2)}}{a_1\tan(-\beta d_2)} \tag{3.10}$$

$$a_1 = \mathfrak{Re}\big(y(-d_1)\big) \tag{3.11}$$

$$b_1 = \mathfrak{Im}\left(\frac{1+b_2+\jmath\tan(-\beta d_2)}{1+\jmath\big(1+b_2\tan(-\beta d_2)\big)}\right) \tag{3.12}$$

$$b = \mathfrak{Im}\big(y(-d_1)\big) \tag{3.13}$$

Si noti che i nomi delle variabili sono stati scelti in modo da semplificare l'esposizione della dimostrazione in 4.3.
Il codice della funzione è:

```
d1=0;
d2=lambda/8;
yl=Zc/Zl;
yd1=(yl + 1i*tan(beta*d1) )/(1 + 1i*yl*tan(beta*d1) ); %
   ammetance in d1 [m] from the load
a1=real(yd1);
b=imag(yd1);

tg=tan(beta*(-d2) ); %element for the next formula
delta=a1*(1+tg^2)-a1^2*tg^2;

%  Note that y1R1 stand for:
% y           1                R              1
% ^           ^                ^              ^
%am.|STUB close load|rigth of the STUB|first sol.

%FIRST SOLUTION
b2=(a1 - sqrt(delta))/(a1*tg); %the imag part of the value
    tha we have to find at the left of the STUB

y2R1=1+1j*b2;
y1L1=(y2R1+1j*tan(-beta*d2))/(1+1j*y2R1*tan(-beta*d2));
b1=imag(y1L1);

bStub11=b1-b; %Ammettance of the stub close to the load
l1R=1/beta*atan(1/(-bStub11));%length of the first STUB
if(l1R<0)
        l1R=l1R+lambda/2;
end
```

```
y1L1=a1+1j*b1; %ammettance at the left of the rigth STUB,
   after the STUB toward the load
y2R1=(y1L1+1j*tan(beta*d2))/(1+1j*y1L1*tan(beta*d2)); %
   ammettance at the rigth of the left STUB, before the
   STUB toward the load

bStub21=-imag(y2R1);
l1L=1/beta*atan(1/(-bStub21));%length of the first STUB
if(l1L<0)
        l1L=l1L+lambda/2;
end
```

Capitolo 4

Dimostrazioni dei risultati

Si riportano qui le dimostrazioni dei risultati usati per trovare le distanze e le lunghezze degli adattatori presentate nel capitolo precedente.

4.1 Adattatore a quarto di lunghezza d'onda

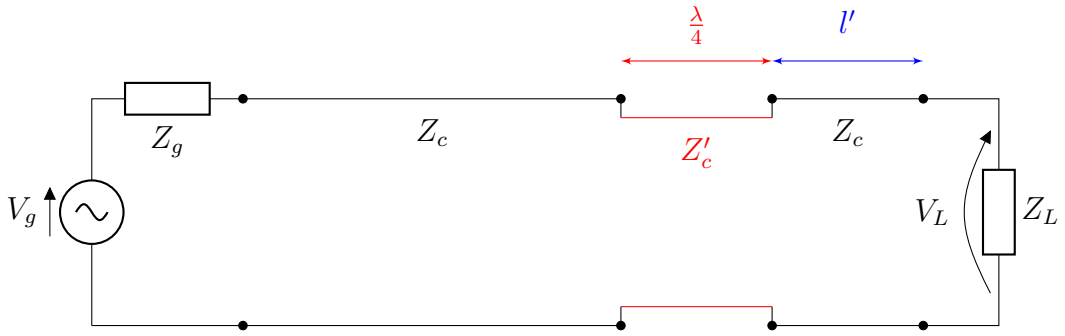

Figura 4.1: Linea di trasmissione con adattatore a quarto di lunghezza d'onda

Questo tipo di adattamento si basa sull'idea di aggiungere un tratto di linea di lunghezza $\frac{\lambda}{4}$ con impedenza Z'_c a un certa distanza dal carico l'.
L'impedenza in un qualsiasi punto della linea a distanza l dal carico è data da (1.4), quindi possiamo scegliere l' t.c. $\mathfrak{Im}(Z(-l')) = 0$
Questo si ha quando è massimo o minimo il modulo della tensione; applicando la soluzione dell'equazione d'onda (1.1) si ha:

$$|V(-l)| = |V_+(1 + \Gamma e^{2j\beta l})| \tag{4.1}$$

da cui troviamo:

$$|V(-l')| = \begin{cases} |1+\Gamma e^{2j\beta l'_1}| & \text{è massimo se} \quad e^{2j\beta l'_1 + j\angle\Gamma} = 1 \Longrightarrow l'_1 = -\frac{\angle\Gamma}{2\beta} \\ |1+\Gamma e^{2j\beta l'_2}| & \text{è minimo se} \quad e^{2j\beta l'_2 + j\angle\Gamma} = -1 \Longrightarrow l'_2 = -\frac{\angle\Gamma}{2\beta} + \frac{\pi}{2\beta} \end{cases} \tag{4.2}$$

Quindi possiamo inserire il tratto di linea a quarto d'onda indifferentemente a distanza l'_1 e l'_2 dal carico.
Procediamo ora a trovare l'impedenza del tratto di linea da aggiungere: sia Z_{in} l'impedenza a distanza l' dal carico, aggiungendo l'adattatore si ha:

$$Z^+(-l') = Z'_c \frac{Z_{in} + \jmath Z_c \tan(\beta\frac{\lambda}{4})}{Z_c + \jmath Z_{in} \tan(\beta\frac{\lambda}{4})} \tag{4.3}$$

Ricordando che $\beta = \frac{2\pi}{\lambda}$ si ha:

$$Z^+(-l') = Z'_c \frac{Z_{in}}{Z_c} \tag{4.4}$$

Siccome vogliamo adattare il carico alla linea, è necessario che $Z^+(-l') = Z_c$, allora si trova:

$$Z'_c = \frac{Z_c^2}{Z_{in}} \tag{4.5}$$

□

4.2 Adattatore a singolo STUB

L'impedenza in un qualsiasi punto della linea a distanza l dal carico vale (1.4), quindi, dal momento che l'ammettenza normalizzata vale $y_{in} = \frac{Z_c}{Z_L}$, si ricava facilmente:

$$y(-l) = \frac{y_L + \jmath \tan(\beta l)}{1 + \jmath y_L \tan(\beta l)} \tag{4.6}$$

Sostituendo $y_l = a + \jmath b$ possiamo riscrivere la relazione come:

$$y(-l) = \frac{a + \jmath b + \jmath \tan(\beta l)}{1 + \jmath(a + \jmath b)\tan(\beta l)} \tag{4.7}$$

Separando la parte reale da quella immaginaria:

$$\begin{aligned} y(-l) &= \frac{a + \jmath b + \jmath \tan(\beta l)}{1 - b\tan(\beta l) + \jmath a \tan(\beta l)} \cdot \frac{1 - b\tan(\beta l) - \jmath a \tan(\beta l)}{1 - b\tan(\beta l) - \jmath a \tan(\beta l)} \\ &= \frac{a + ab\tan^2(\beta l)}{1 - 2b\tan(\beta l) + b^2\tan^2(\beta l) + a^2\tan^2(\beta l)} \\ &\quad + \jmath \frac{b + \tan(\beta l) - b^2 - b\tan^2(\beta l) - a^2\tan(\beta l)}{1 - 2b\tan(\beta l) + b^2\tan^2(\beta l) + a^2\tan^2(\beta l)} \end{aligned} \tag{4.8}$$

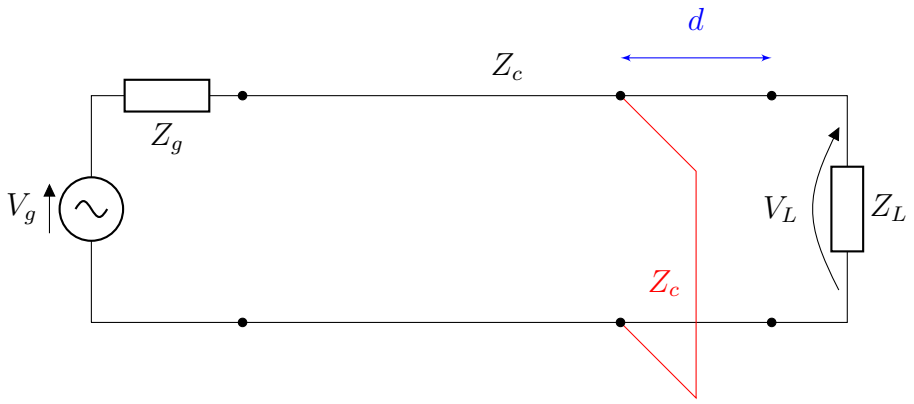

Figura 4.2: Linea di trasmissione con adattatore a singolo STUB

Uno STUB di impedenza Z_c pari a quella della linea ha ammettenza normalizzata $y_{STUB} = -\jmath \cdot \frac{1}{\tan(\beta l)}$. Prendiamo un punto a distanza d dal carico t.c. $y(-d) = 1 + \jmath b$ (con $b \in \Re$). Aggiungendo uno STUB in parallelo alla linea, alla distanza d dal carico, si ha: $y_{-d} = y_{STUB} + y(-d) = b_{STUB} + 1 + b(-d)$. Scegliendo la lunghezza dello STUB t.c. $b_{STUB} = b(-d)$, si ha $y_{-d} = 1$, ovvero la condizione di linea adattata.
Il nostro scopo è ora quello di trovare tale d, ovvero l t.c.:

$$\frac{a + ab\tan^2(\beta l)}{1 - 2b\tan(\beta l) + b^2\tan^2(\beta l) + a^2\tan^2(\beta l)} = 1 \tag{4.9}$$

di conseguenza:

$$\tan^2(\beta l) \cdot (b^2 + a^2 - a) - 2b\tan(\beta l) + 1 - a = 0 \tag{4.10}$$

$$\tan(\beta l) = \frac{2b \pm \sqrt{4b^2 - 4(b^2 + a^2 - a) \cdot (1 - a)}}{2 \cdot (b^2 + a^2 - a)} \tag{4.11}$$

quindi i punti in cui porre lo STUB saranno dati da:

$$d = \frac{1}{\beta} \cdot \arctan \frac{2b \pm \sqrt{4b^2 - 4(b^2 + a^2 - a) \cdot (1 - a)}}{2 \cdot (b^2 + a^2 - a)} + const \cdot \frac{\lambda}{2} \tag{4.12}$$

dove λ è la lunghezza d'onda, mentre $const \in \mathbb{N}$ e $const$ è t.c. $d \geq 0$.
La lunghezza dello STUB invece deve essere t.c.:

$$y_{STUB} = -\jmath \cdot \frac{1}{\tan(\beta l_1)} = -\mathfrak{Im}\big(y(-d)\big) \tag{4.13}$$

dove $y(-d) = \frac{y_L + \jmath\tan(\beta d)}{1 + \jmath y_L \tan(\beta d)}$, quindi:

$$l_1 = \frac{1}{\beta} \cdot \arctan\left(-\frac{1}{\mathfrak{Im}\big(y(-d)\big)} \right) + N \cdot \frac{\lambda}{2} \tag{4.14}$$

In conclusione:

$$\begin{cases} d = \frac{1}{\beta} \cdot \arctan \frac{2b \pm \sqrt{4b^2 - 4(b^2+a^2-a)\cdot(1-a)}}{2\cdot(b^2+a^2-a)} + N \cdot \frac{\lambda}{2} \\ l_1 = \frac{1}{\beta} \cdot \arctan\left(-\frac{1}{\Im\mathfrak{m}\big(y(-d)\big)} \right) \end{cases} \tag{4.15}$$

dove $N \in \mathbb{N}$.

□

4.3 Adattatore a doppio STUB

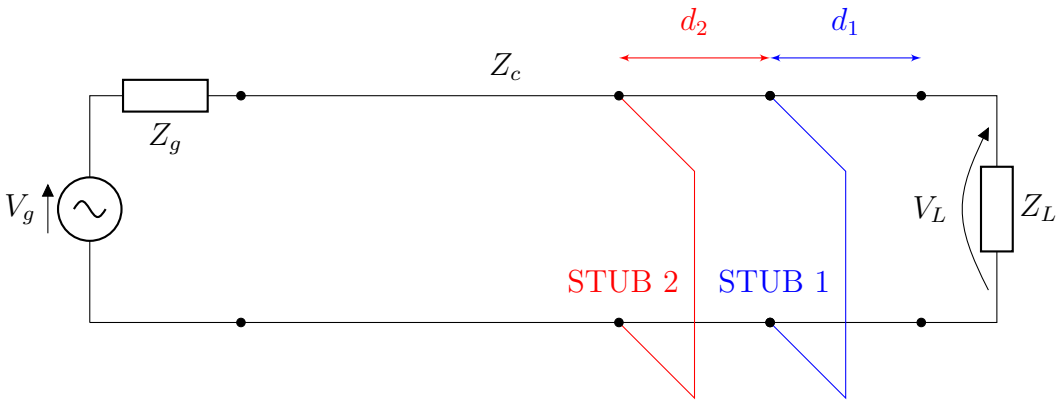

Figura 4.3: Linea di trasmissione con adattatore a doppio STUB

Sia data una linea di trasmissione di ammettenza Y_c e carico di ammettenza normalizzata y_L che vogliamo adattare tramite due STUB posti a distanza d_1 e $d_1 + d_2$ dal carico (ved. figura 4.3), con d_1 e d_2 scelte a priori. Si vuole trovare la lunghezza che dovranno avere i due STUB realizzati tramite cortocircuito tali che rendano la linea adattata.
Si noti che per comprendere meglio la dimostrazione può essere utile seguire i passaggi nella carta di Smith riportata nella figura 4.4.
Chiamiamo $y_{circ}(0)$ l'insieme dei valori t.c. $\mathfrak{Re}(y(-l)) = 1$ (con l lunghezza generica). Abbiamo quindi che $y_{circ}(0) = 1 + \jmath b_2$*, con $b_2 \in \Re$. Supponendo di spostarci sulla linea a distanza d_2 verso il generatore, chiamiamo l'ammettenza di quel punto $y_{circ}(+d_2)$†. Dalla (4.6) e dalla definizione di $y_{circ}(0)$ troviamo che:

$$y_{circ}(+d_2) = \frac{y_{circ}(0) + \jmath \tan(-\beta d_2)}{1 + \jmath y_{circ}(0) \tan(-\beta d_2)} = \frac{1 + \jmath\big(b_2 + \tan(-\beta d_2)\big)}{1 - b_2 \tan(-\beta d_2) + \jmath \tan(-\beta d_2)} \tag{4.16}$$

*$y_{circ}(0)$ è l'ammettenza che vogliamo avere a distanza $d_1 + d_2$ dal carico dopo l'adattamento.
†$y_{circ}(+d_2)$ è l'ammettenza che vogliamo avere a distanza d_1 dal carico dopo l'adattamento, ovviamente qui ci stiamo spostando verso il carico, quindi dobbiamo mettere il "+".

Separando la parte immaginaria da quella reale troviamo:

$$\begin{aligned} y_{circ}(+d_2) &= \frac{1+\jmath\big(b_2+\tan(-\beta d_2)\big)}{1-b_2\tan(-\beta d_2)+\jmath\tan(-\beta d_2)}\cdot\frac{1-b_2\tan(-\beta d_2)+\jmath\tan(-\beta d_2)}{1-b_2\tan(-\beta d_2)+\jmath\tan(-\beta d_2)} \\ &= \frac{1+\tan^2(-\beta d_2)}{1-2b_2\tan(-\beta d_2))+b_2^2\tan^2(-\beta d_2)+\tan^2(-\beta d_2)} \\ &+\jmath\frac{b_2+2\tan(-\beta d_2)-b_2^2\tan(-\beta d_2)-b_2\tan^2(-\beta d_2)}{1-2b_2\tan(-\beta d_2)+b_2^2\tan^2(-\beta d_2)+\tan^2(-\beta d_2)} \end{aligned} \tag{4.17}$$

Ricordando la (4.6), possiamo scrivere l'ammettenza a distanza d_1 dal carico come:

$$y(-d_1) = \frac{y_L+\jmath\tan(\beta d_1)}{1+\jmath y_L\tan(\beta d_1)} \tag{4.18}$$

Scriviamo ora $y(-d_1) = a_1 + \jmath b$ e affinché la linea sia adattata vogliamo b t.c. $a_1 = \mathfrak{Re}(y_{circ}(+d_2))$, quindi:

$$a_1 = \frac{1+\tan^2(-\beta d_2)}{1-2b_2\tan(-\beta d_2)+b_2^2\tan^2(-\beta d_2)+\tan^2(-\beta d_2)} \tag{4.19}$$

si ricava quindi:

$$b_2^2 a_1\tan(-\beta d_2))^2 - b_2 2a_1\tan(-\beta d_2) + (a_1-1)(1+\tan^2(-\beta d_2)) = 0 \tag{4.20}$$

che è un'equazione di secondo grado, le cui soluzioni sono:

$$b_2 = \frac{2a_1\tan(-\beta d_2) \pm \sqrt{4a_1^2\tan^2(-\beta d_2) - 4a_1\tan^2(-\beta d_2)(a_1-1)(1+\tan^2(-\beta d_2))}}{a_1\tan^2(-\beta d_2)} \tag{4.21}$$

$$= \frac{a_1 \pm \sqrt{a_1(1+\tan^2(-\beta d_2)) - a_1^2\tan^2(-\beta d_2)}}{a_1\tan(-\beta d_2)} \tag{4.22}$$

Quindi l'ammettenza normalizzata che dovrà realizzare lo STUB 2 è $b_{STUB2} = b - b_2$ [‡], da (4.14) si ha che la lunghezza dello STUB 2 sarà:

$$l_2 = \frac{1}{\beta}\cdot\arctan\Big(-\frac{1}{b_{STUB2}}\Big) \tag{4.23}$$

[‡]Dal momento che lo STUB è inserito in parallelo le parti immaginarie delle ammetenze si sommano, ovvero: $b = b_{STUB2} + b_2$.

Inserendo lo STUB 2 si ha subito a destra di esso un'impedenza $y_{-d_1-d_2} = 1 + \jmath b_2$, quindi per la (4.6) si ha:

$$y(-d_1) = \frac{y_{-d_1-d_2} + \jmath \tan(-\beta d_2)}{1 + \jmath(y_{-d_1-d_2}) \tan(-\beta d_2)} \tag{4.24}$$

Prendendo $b_1 = \mathfrak{Im}(y(-d_1))$[§], si ha $b_{STUB1} = -b_1$ e quindi per (4.14):

$$l_1 = \frac{1}{\beta} \cdot \arctan\Big(-\frac{1}{b_{STUB1}}\Big) \tag{4.25}$$

In conclusione:

$$\begin{cases} b_2 = \frac{a_1 \pm \sqrt{a_1(1+\tan^2(-\beta d_2)) - a_1^2 \tan^2(-\beta d_2)}}{a_1 \tan(-\beta d_2)} \\ l_2 = \frac{1}{\beta} \cdot \arctan\Big(-\frac{1}{b - b_2}\Big) \\ b_1 = \mathfrak{Im}\Big(\frac{1 + b_2 + \jmath \tan(-\beta d_2)}{1 + \jmath(1 + b_2 \tan(-\beta d_2)}\Big) \\ l_1 = \frac{1}{\beta} \cdot \arctan\Big(-\frac{1}{-b_1}\Big) \end{cases} \tag{4.26}$$

□

[§]Rispetto alla carta di Smith b_1 può essere ottenuto ruotando $1 + b_2$ di d_2 verso il carico.

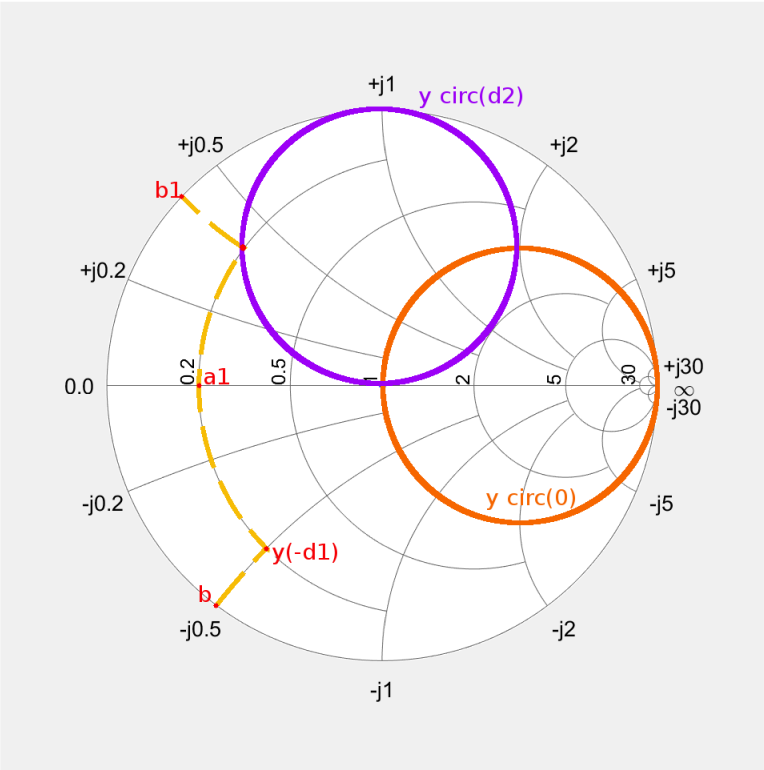

Figura 4.4: Adattatamento a doppio STUB con $d_1 = 0$ e $d_2 = \lambda/8$ tramite carta di Smith

Capitolo 5

Conclusioni

L'applicativo Matlab sviluppato è capace di risolvere correttamente problemi relativi alle linee di trasmissione, rispettando le specifiche date.
Ogni funzione è stata testata sviluppando manualmente i calcoli e verificando che i risultati ottenuti dall'applicazione coincidano con quelli teorici. Col fine di mostrare come si sono ottenuti i risultati implementati all'interno della tesi è presente la dimostrazione di tutte le relazioni usate per dimensionare gli adattatori, il resto si può verificare in modo banale.
Si noti che prima dello sviluppo dell'applicazione è stata fatta un'analisi delle soluzioni software già esistenti. Nonostante esistano già diverse applicazioni simili si è cercato di dare un taglio più didattico al programma, mostrando come variano nello spazio grandezze come tensione e corrente. L'applicativo implementa inoltre la possibilità di dimensionare un adattatore a doppio STUB, caratteristica che è raramente implementata.

Appendice A

Il codice

Viene qui riportato il codice Matlab dell'applicazione. Si noti che per verificarlo è sufficiente copiarlo in Matlab salvando il file come

transmission_Line_Solver

.

```
classdef transmission_Line_Solver13ProfCorrections <
   matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                        matlab.ui.Figure
        GridLayout                      matlab.ui.container.
           GridLayout
        LeftPanel                       matlab.ui.container.
           Panel
        PlotsettingsLabel               matlab.ui.control.
           Label
        showingDropDown                 matlab.ui.control.
           DropDown
        showingDropDownLabel            matlab.ui.control.
           Label
        SelectListBox                   matlab.ui.control.
           ListBox
        SelectListBoxLabel              matlab.ui.control.
           Label
        ImpedanceMatchingLabel          matlab.ui.control.
           Label
```

```
        ZcOhmEditField                  matlab.ui.control.
           EditField
        ZcOhmEditFieldLabel             matlab.ui.control.
           Label
        ZloadOhmEditField               matlab.ui.control.
           EditField
        ZloadOhmEditFieldLabel          matlab.ui.control.
           Label
        CalculateButton                 matlab.ui.control.
           Button
        freqMHzEditField                matlab.ui.control.
           NumericEditField
        freqMHzEditFieldLabel           matlab.ui.control.
           Label
        VgVEditField                    matlab.ui.control.
           NumericEditField
        VgVEditFieldLabel               matlab.ui.control.
           Label
        LinelengthmmEditField           matlab.ui.control.
           NumericEditField
        LinelengthmmEditFieldLabel      matlab.ui.control.
           Label
        OtherLabel                      matlab.ui.control.
           Label
        ImpedanceRealiImaginaryLabel    matlab.ui.control.
           Label
        RightPanel                      matlab.ui.container.
           Panel
        Output2EditField                matlab.ui.control.
           EditField
        Output2EditFieldLabel           matlab.ui.control.
           Label
        Output1EditField                matlab.ui.control.
           EditField
        Output1EditFieldLabel           matlab.ui.control.
           Label
        UIAxes2                         matlab.ui.control.
           UIAxes
        UIAxes                          matlab.ui.control.
           UIAxes
    end
```

```
% Properties that correspond to apps with auto-reflow
properties (Access = private)
    onePanelWidth = 576;
end

%Program methods
methods (Access = private)

    function results = printing(app, message1,
       message2)
        app.Output1EditField.Value=message1
        app.Output2EditField.Value=message2
    end

    function sNum = engn(app, value) %Engineering
       notation
        exp=0;
        if(abs(value) == 0)
            exp=0;
        else
            exp= floor(log10(abs(value)));
        end
        if ( (exp < 3) && (exp >=0) )
            exp = 0; % Display without exponent
        else
            while (mod(exp, 3))
                exp= exp - 1;
            end
        end
        frac=value/(10^exp); % Adjust fraction to
           exponent
        if (exp == 0)
            sNum = sprintf('%G', frac);
        else
            sNum = sprintf('%GE%+.2d', frac, exp);
        end
    end

    function sNum = engn1digits(app, value) %
       Engineering notation
        exp=0;
        if(abs(value) == 0)
```

```
            exp=0;
        else
            exp= floor(log10(abs(value)));
        end
        if ( (exp < 3) && (exp >=0) )
            exp = 0; % Display without exponent
        else
            while (mod(exp, 3))
                exp= exp - 1;
            end
        end
        frac=value/(10^exp); % Adjust fraction to
           exponent
        if (exp == 0)
            sNum = sprintf('%.1f', frac);
        else
            sNum = sprintf('%GE%+.2d', frac, exp);
        end
    end

    function plotting(app, select, V0p, beta, gamma,
       Zc, z)

        switch select
            case "plot Voltage module"
                V = abs(V0p.*( exp(-1j*beta.*z )+ (
                   gamma*exp(1j*beta.*z ))) );
                title(app.UIAxes, "Voltage module")
                ylabel(app.UIAxes, "|V(z)| [V]")
                plot(app.UIAxes, z, V)

            case "plot Current module"
                V = abs((V0p/Zc).*( exp(-1j*beta.*z )-
                    (gamma*exp(1j*beta.*z ))) );
                title(app.UIAxes, "Current module")
                ylabel(app.UIAxes, "|I(z)| [A]")
                plot(app.UIAxes, z, V)

            case "plot Voltage phase"
                V = angle(V0p.*( exp(-1j*beta.*z )+ (
                   gamma*exp(1j*beta.*z ))) );
                V=unwrap(V);
```

```
                title(app.UIAxes, "Voltage phase")
                ylabel(app.UIAxes, "p(V(z)) [rad]")
                plot(app.UIAxes, z, V)

            case "plot Current phase"
                V = angle((V0p/Zc).*( exp(-1j*beta.*z
                    )- (gamma*exp(1j*beta.*z ))) );
                V=unwrap(V);
                title(app.UIAxes, "Current phase")
                ylabel(app.UIAxes, "p(I(z)) [rad]")
                plot(app.UIAxes, z, V)

            case "Plot Voltage real part"
                V = real( V0p.*( exp(-1j*beta.*z )+ (
                    gamma*exp(1j*beta.*z ))) );
                title(app.UIAxes, "Voltage real part")
                ylabel(app.UIAxes, "V(z) [V]")
                plot(app.UIAxes, z, V)

            case "Plot Voltage imaginary part"
                V = imag( V0p.*( exp(-1j*beta.*z )+ (
                    gamma*exp(1j*beta.*z ))) );
                title(app.UIAxes, "Voltage imaginary
                    part")
                ylabel(app.UIAxes, "V(z) [V]")
                plot(app.UIAxes, z, V)

        end

    end


end


% Callbacks that handle component events
methods (Access = private)

    % Button pushed function: CalculateButton
    function CalculateButtonPushed(app, event)

        %Data aquisition
```

```
Lmax= app.LinelengthmmEditField.Value;
f = app.freqMHzEditField.Value*10^6;
Zc = str2double(app.ZcOhmEditField.Value);
Zl = str2double(app.ZloadOhmEditField.Value);
Vg = app.VgVEditField.Value;    %V+ voltage in
    the load
V0p=Vg/2;                       %V+=Vg/2,
   supposing Zg=Zc

epsilonr=1; %permettivita' relativa
lambda=physconst('LightSpeed')/f*1/sqrt(
   epsilonr);
    %Version 1: code to show the full length
       asked
    Lmax=Lmax*0.001;%meter to mm

    %Version 2: code to show the full length
       asked in unit of
    %lambda
    %Lmax=Lmax*lambda;

    %Version 3: code to show only one labda
       length unit in the axes
    %LambdaLength=Lmax/lambda;
    %LambdaLength=LambdaLength-floor(
       LambdaLength);
                                %take the
                                   fractional
                                   part of
                                   LambdaLength

    %Lmax=LambdaLength*lambda+lambda;
    %now Lamx contains only one labda
       oscillation

Lc=Lmax/1000;
z=-Lmax:Lc:0;

%Data calculation
omega= 2*pi*f;
beta=omega*sqrt(epsilonr)/physconst('
   LightSpeed');
```

```
            gamma = z2gamma(Zl,Zc);

            %Metching
            select = app.SelectListBox.Value;
            message1="None"; %message to print
            message2="None";
            switch select


%% ___________Lambda/4 adapter___________

case "Lambda / 4"   %Calculate Impedence of the line to
   add and the distance from the load to put the line in
   the maxium point of voltage
         dist1=angle(gamma)/(2*beta); %In meter, dist from
            the first maximum
      if(dist1<0)
         dist1=dist1+lambda/2;
      end
      dist2=dist1+lambda/4; %In meter, dist from the first
         maximum
      if(dist2>lambda/2)
         dist2=dist2-lambda/2;
      end
      S=(1+abs(gamma))/(1-abs(gamma)); %ROS
      Zc21=Zc*sqrt(S);     %In the maximum of voltage
      Zc22=Zc/sqrt(S);     %In the minimum of voltage

      Zc21=engn(app, Zc21);
      Zc22=engn(app, Zc22);

      dist1=engn(app, dist1*1000); %Use millimiter
      dist2=engn(app, dist2*1000);
      message1=strcat("Adapter impedence of ",Zc21," [Ohm]
         at a distance of ",dist1, " [mm] from the load");
      message2=strcat("Or adapter impedence of ",Zc22," [
         Ohm] at a distance of ",dist2, " [mm] from the
         load");


%% ___________Single stub adapter___________
```

```
case "S. stub"
       %Single stub matching with shorted line
      a=real(Zc/Zl);
      b=imag(Zc/Zl);
      yl=Zc/Zl;    %load ammetance
      d1=0;
      if(real(Zc/Zl) ~= 1 && real(Zl/Zc) ~= 1)
         d1=1/beta*atan((2*b+sqrt(4*b^2-4*(b^2+a^2-a)*(1-a)
            ) )/(2*(b^2+a^2-a)));
      end
      if(real(Zc/Zl) ~= 1 && real(Zl/Zc) == 1)
          d1=lambda/4+lambda*0.176;
      end
      if(d1<0)
         d1=d1+lambda/2;
      end
      %distance from the first point with RE{Y(l11)}=RE{Yc}
          [m]
      d1print=engn1digits(app, d1*1000);
      b1=imag((yl+1i*tan(beta*d1))/(1+1i*tan(beta*d1)*yl));
           %immaginary part at the left of the stub
      bstub1=-b1;  %ammetance that the stub must create
      l1=1/beta*atan(-1/bstub1);
      if(l1<0)
         l1=l1+lambda/2;
      end
      l1print=engn1digits(app, l1*1000);

      d2=lambda/2;
      if(real(Zc/Zl) ~= 1 && real(Zl/Zc) ~= 1)
         d2=1/beta*atan((2*b-sqrt(4*b^2-4*(b^2+a^2-a)*(1-a)
            ) )/(2*(b^2+a^2-a)));
      end
      if(real(Zc/Zl) ~= 1 && real(Zl/Zc) == 1)
         d2=lambda/4;
      end
      if(d2<0)
              d2=d2+lambda/2;
      end
      %distance from the second point with RE{Y(l11)}=RE{Yc
         } [m]
      d2print=engn1digits(app, d2*1000);
```

```
    b2=imag((yl+1i*tan(beta*d2))/(1+1i*tan(beta*d2)*yl));
        %immaginary part at the left of the stub
    bstub2=-b2;  %ammetance that the stub must create
    l2=1/beta*atan(-1/bstub2);
    if(l2<0)
        l2=l2+lambda/2;
    end
    l2print=engn1digits(app, l2*1000);

    message1=strcat("Stub be placed ",d1print," [mm] from
       the load and ",l1print, " [mm] length");
    message2=strcat("Or stub be placed ",d2print," [mm]
       from the load and ",l2print, " [mm] length");

        %Testing: code for the test of the function
        %message2=strcat(message2, ", The real part of the
            nomalized ammitance at these distances is: ",
           testSStub(d1), " and ", testSStub(d2));


%% ___________Double stub adapter___________
case "D. stub"
        %Double stub matching with 2 shorted line plaved
           at a fixed distance
     d1=0;
     d2=lambda/8;

     yl=Zc/Zl;
     yd1=(yl + 1i*tan(beta*d1) )/(1 + 1i*yl*tan(beta*d1) )
        ; %ammetance in d1 [m] from the load
     a1=real(yd1);
     b=imag(yd1);

     tg=tan(beta*(-d2) ); %element for the next formula
     delta=a1*(1+tg^2)-a1^2*tg^2;

%Note that y1R1 stand for:
%      y           1                        R
                                 1
%     ^           ^                        ^
                                 ^
```

```
% ammetance   stub close to the load   at the rigth of the
    stub  first solution

%FIRST SOLUTION
    b2=(a1 - sqrt(delta))/(a1*tg); %the imag part of the
       value tha we have to find at the left of the stub

     y2R1=1+1j*b2;
     y1L1=(y2R1+1j*tan(-beta*d2))/(1+1j*y2R1*tan(-beta*d2)
        );
     b1=imag(y1L1);

     bStub11=b1-b; %Ammettance of the stub close to the
        load
     l1R=1/beta*atan(1/(-bStub11));%length of the first
        stub
     if(l1R<0)
          l1R=l1R+lambda/2;
     end
     l11=engn1digits(app, l1R*1000);

     y1L1=a1+1j*b1; %ammettance at the left of the rigth
        stub, after the stub toward the load
     y2R1=(y1L1+1j*tan(beta*d2))/(1+1j*y1L1*tan(beta*d2));
                                              %ammettance
                                                 at the
                                                 rigth of
                                                 tyhe left
                                                 stub,
                                                 before the
                                                  stub
                                                 toward the
                                                  load

     bStub21=-imag(y2R1);
     l1L=1/beta*atan(1/(-bStub21));%length of the first
        stub
     if(l1L<0)
            l1L=l1L+lambda/2;
     end
     l12=engn1digits(app, l1L*1000);
```

```
%SECOND SOLUTION
      b2=(a1 + sqrt(delta))/(a1*tg);  %the imag part of the
          value tha we have to find at the left of the stub

      y2R2=1+1j*b2;
      y1L2=(y2R2+1j*tan(-beta*d2))/(1+1j*y2R2*tan(-beta*d2)
         );
      b1=imag(y1L2);

      bStub12=b1-b; %Ammettance of the stub close to the
         load
      l2R=1/beta*atan(1/(-bStub12));%length of the first
         stub
      if(l2R<0)
           l2R=l2R+lambda/2;
      end
      l21=engn1digits(app, l2R*1000);

      y1L2=a1+1j*b1; %ammettance at the left of the rigth
         stub, after the stub toward the load
      y2R2=(y1L2+1j*tan(beta*d2))/(1+1j*y1L2*tan(beta*d2));
      %ammettance at the rigth of tyhe left stub, before
         the stub toward the load

      %bStub22=-imag(y2R2)
      bStub22=-b2;
      l2L=1/beta*atan(1/(-bStub22));%length of the first
         stub
      if(l2L<0)
           l2L=l2L+lambda/2;
      end
      l22=engn1digits(app, l2L*1000);

      message1=strcat("First solution: the lengh of the
         stub close to the load is ",l11, " [mm], the
         second stub length is ",l12, " [mm]");

      %Testing
      %y1r=engn(app, real(y2R1));
      %y1i=engn(app, imag(y2R1));
      %message1=strcat(message1, ", The impedence is ", y1r
         , " +j ", y1i);
```

```
        message2=strcat("Second solution: the lengh of the
            stub close to the load is ",l21, " [mm], the
            second stub length is ",l22, " [mm]");

        %Testing
        %y2r=engn(app, real(y2R2));
        %y2i=engn(app, imag(y2R2));
        %message2=strcat(message2, ", The impedence is ", y2r
            , " +j ", y2i);
end


%% ___________PRINTING___________

        %Output text
        printing(app, message1, message2)

        %Plotting on axes chart
        gamma = z2gamma(Zl,Zc);              %reflection
            coeffint
        select = app.showingDropDown.Value;
        plotting(app, select, V0p, beta, gamma, Zc, z)

        %plot on smith chrt
        %print on chart under the voltage chart
        s1 = smithplot(app.UIAxes2, gamma,'Color','b','
            LineStyle','-.','LineWidth',3);
        s1.Marker = {'+'};

        %print on a new figure
        figure
        gamma = z2gamma(Zl+0.0001*1j,Zc); % for printing an
            imaginary number
        s2 = smithplot(gamma,'Color','b','LineStyle','-.','
            LineWidth',3);
        s2.Marker = {'+'} ;


%% ___________Program I/O Function___________

        function out = testSStub(d) %Function for testing
```

```
    the single stub matching, return the yin after
    the stub(must be 1 for correct matching)
     yl=Zc/Zl;
     yin=(yl+1i*tan(beta*d))/(1+1i*yl*tan(beta*d));
     out=engn(app, real(yin));
end

end

% Value changed function: SelectListBox
function SelectListBoxValueChanged(app, event)
    value = app.SelectListBox.Value;

    message="No metching selected"; %message to
       print
    switch value

        case "Lambda / 4"

            message="lambda/4 matching slected";

        case "S. stub"
            message="single stub matching slected
               with short line";

        case "D. stub"
            message="double stub matching slected.
                !Not all the line can be DS
               matched";
    end
    printing(app, message, "None")
    %CalculateButtonPushed(app, event)
end

% Value changed function: freqMHzEditField
function freqMHzEditFieldValueChanged(app, event)
    f = app.freqMHzEditField.Value;     %frequency
        in MHz
    f=f*10^6;                           %frequency
        in Hz
    epsilonr=1;                         %
       permettivita' relativa
```

```
        lambda=physconst('LightSpeed')/f*1/sqrt(
            epsilonr);   %wave length [m]

        lambda=engn(app, lambda*1000);
        message=strcat("The wave length is: ",lambda,
            " [mm]");
        printing(app, message, "None")
    end

    % Value changed function: ZcOhmEditField,
        ZloadOhmEditField
    function ZcOhmEditFieldValueChanged(app, event)
        Zc = str2double(app.ZcOhmEditField.Value);
        Zl = str2double(app.ZloadOhmEditField.Value);
        gamma = z2gamma(Zl,Zc);
        SWR=(1+abs(gamma))/(1-abs(gamma));
        %gammaR=engn(app, real(gamma));
        %gammaI=engn(app, imag(gamma));
        gammaM=engn(app, abs(gamma));
        gammaP=engn(app, angle(gamma));
        SWR=engn(app, SWR);
        %message1=strcat("The reflection cofficent on
            the load is: ",gammaR, " + i",gammaI);
        txt='angle';
        message1=strcat("The reflection cofficent on
            the load is: ",gammaM, " ",txt,gammaP);
        message2=strcat("The SWR is: ",SWR);
        printing(app, message1, message2)
    end

    % Value changed function: showingDropDown
    function showingDropDownValueChanged(app, event)
        CalculateButtonPushed(app, event)

    end

    % Changes arrangement of the app based on UIFigure
        width
    function updateAppLayout(app, event)
        currentFigureWidth = app.UIFigure.Position(3);
        if(currentFigureWidth <= app.onePanelWidth)
            % Change to a 2x1 grid
```

```
            app.GridLayout.RowHeight = {569, 569};
            app.GridLayout.ColumnWidth = {'1x'};
            app.RightPanel.Layout.Row = 2;
            app.RightPanel.Layout.Column = 1;
        else
            % Change to a 1x2 grid
            app.GridLayout.RowHeight = {'1x'};
            app.GridLayout.ColumnWidth = {223, '1x'};
            app.RightPanel.Layout.Row = 1;
            app.RightPanel.Layout.Column = 2;
        end
    end
end

% Component initialization
methods (Access = private)

    % Create UIFigure and components
    function createComponents(app)

        % Create UIFigure and hide until all
           components are created
        app.UIFigure = uifigure('Visible', 'off');
        app.UIFigure.AutoResizeChildren = 'off';
        app.UIFigure.Position = [100 100 661 569];
        app.UIFigure.Name = 'MATLAB App';
        app.UIFigure.SizeChangedFcn =
           createCallbackFcn(app, @updateAppLayout,
           true);

        % Create GridLayout
        app.GridLayout = uigridlayout(app.UIFigure);
        app.GridLayout.ColumnWidth = {223, '1x'};
        app.GridLayout.RowHeight = {'1x'};
        app.GridLayout.ColumnSpacing = 0;
        app.GridLayout.RowSpacing = 0;
        app.GridLayout.Padding = [0 0 0 0];
        app.GridLayout.Scrollable = 'on';

        % Create LeftPanel
        app.LeftPanel = uipanel(app.GridLayout);
        app.LeftPanel.Layout.Row = 1;
```

```
app.LeftPanel.Layout.Column = 1;

% Create ImpedanceRealiImaginaryLabel
app.ImpedanceRealiImaginaryLabel = uilabel(app
    .LeftPanel);
app.ImpedanceRealiImaginaryLabel.Position =
    [44 533 173 22];
app.ImpedanceRealiImaginaryLabel.Text = '
    Impedance [Real + i*Imaginary]';

% Create OtherLabel
app.OtherLabel = uilabel(app.LeftPanel);
app.OtherLabel.Position = [41 428 36 22];
app.OtherLabel.Text = 'Other';

% Create LinelengthmmEditFieldLabel
app.LinelengthmmEditFieldLabel = uilabel(app.
    LeftPanel);
app.LinelengthmmEditFieldLabel.
    HorizontalAlignment = 'right';
app.LinelengthmmEditFieldLabel.Position = [9
    395 94 22];
app.LinelengthmmEditFieldLabel.Text = 'Line
    length [mm]';

% Create LinelengthmmEditField
app.LinelengthmmEditField = uieditfield(app.
    LeftPanel, 'numeric');
app.LinelengthmmEditField.Position = [117 395
    100 22];
app.LinelengthmmEditField.Value = 500;

% Create VgVEditFieldLabel
app.VgVEditFieldLabel = uilabel(app.LeftPanel)
    ;
app.VgVEditFieldLabel.HorizontalAlignment = '
    right';
app.VgVEditFieldLabel.Position = [58 359 44
    22];
app.VgVEditFieldLabel.Text = '|Vg| [V]';

% Create VgVEditField
```

```
app.VgVEditField = uieditfield(app.LeftPanel,
    'numeric');
app.VgVEditField.Position = [117 359 100 22];
app.VgVEditField.Value = 10;

% Create freqMHzEditFieldLabel
app.freqMHzEditFieldLabel = uilabel(app.
    LeftPanel);
app.freqMHzEditFieldLabel.HorizontalAlignment
    = 'right';
app.freqMHzEditFieldLabel.Position = [41 323
    61 22];
app.freqMHzEditFieldLabel.Text = 'freq [MHz]';

% Create freqMHzEditField
app.freqMHzEditField = uieditfield(app.
    LeftPanel, 'numeric');
app.freqMHzEditField.ValueChangedFcn =
    createCallbackFcn(app, @
    freqMHzEditFieldValueChanged, true);
app.freqMHzEditField.Position = [117 323 100
    22];
app.freqMHzEditField.Value = 1000;

% Create CalculateButton
app.CalculateButton = uibutton(app.LeftPanel,
    'push');
app.CalculateButton.ButtonPushedFcn =
    createCallbackFcn(app, @
    CalculateButtonPushed, true);
app.CalculateButton.Position = [84 23 100 22];
app.CalculateButton.Text = 'Calculate';

% Create ZloadOhmEditFieldLabel
app.ZloadOhmEditFieldLabel = uilabel(app.
    LeftPanel);
app.ZloadOhmEditFieldLabel.HorizontalAlignment
     = 'right';
app.ZloadOhmEditFieldLabel.Position = [30 467
    72 22];
app.ZloadOhmEditFieldLabel.Text = 'Zload [Ohm]
    ';
```

```
% Create ZloadOhmEditField
app.ZloadOhmEditField = uieditfield(app.
    LeftPanel, 'text');
app.ZloadOhmEditField.ValueChangedFcn =
    createCallbackFcn(app, @
    ZcOhmEditFieldValueChanged, true);
app.ZloadOhmEditField.Position = [117 467 100
    22];
app.ZloadOhmEditField.Value = '50';

% Create ZcOhmEditFieldLabel
app.ZcOhmEditFieldLabel = uilabel(app.
    LeftPanel);
app.ZcOhmEditFieldLabel.HorizontalAlignment =
    'right';
app.ZcOhmEditFieldLabel.Position = [47 502 55
    22];
app.ZcOhmEditFieldLabel.Text = 'Zc [Ohm]';

% Create ZcOhmEditField
app.ZcOhmEditField = uieditfield(app.LeftPanel
    , 'text');
app.ZcOhmEditField.ValueChangedFcn =
    createCallbackFcn(app, @
    ZcOhmEditFieldValueChanged, true);
app.ZcOhmEditField.Position = [117 502 100
    22];
app.ZcOhmEditField.Value = '50';

% Create ImpedanceMatchingLabel
app.ImpedanceMatchingLabel = uilabel(app.
    LeftPanel);
app.ImpedanceMatchingLabel.Position = [44 277
    117 22];
app.ImpedanceMatchingLabel.Text = 'Impedance
    Matching';

% Create SelectListBoxLabel
app.SelectListBoxLabel = uilabel(app.LeftPanel
    );
app.SelectListBoxLabel.HorizontalAlignment = '
```

```
    right';
app.SelectListBoxLabel.Position = [63 243 42
    22];
app.SelectListBoxLabel.Text = 'Select:';

% Create SelectListBox
app.SelectListBox = uilistbox(app.LeftPanel);
app.SelectListBox.Items = {'None', 'Lambda / 4
    ', 'S. stub', 'D. stub'};
app.SelectListBox.ValueChangedFcn =
    createCallbackFcn(app, @
    SelectListBoxValueChanged, true);
app.SelectListBox.Position = [117 169 100 98];
app.SelectListBox.Value = 'None';

% Create showingDropDownLabel
app.showingDropDownLabel = uilabel(app.
    LeftPanel);
app.showingDropDownLabel.HorizontalAlignment =
     'right';
app.showingDropDownLabel.Position = [37 104 50
     22];
app.showingDropDownLabel.Text = 'showing';

% Create showingDropDown
app.showingDropDown = uidropdown(app.LeftPanel
    );
app.showingDropDown.Items = {'plot Voltage
    module', 'plot Current module', 'plot
    Voltage phase', 'plot Current phase', 'Plot
     Voltage real part', 'Plot Voltage
    imaginary part'};
app.showingDropDown.ValueChangedFcn =
    createCallbackFcn(app, @
    showingDropDownValueChanged, true);
app.showingDropDown.Position = [102 104 100
    22];
app.showingDropDown.Value = 'plot Voltage
    module';

% Create PlotsettingsLabel
app.PlotsettingsLabel = uilabel(app.LeftPanel)
```

```
    ;
app.PlotsettingsLabel.Position = [42 135 71
    22];
app.PlotsettingsLabel.Text = 'Plot settings';

% Create RightPanel
app.RightPanel = uipanel(app.GridLayout);
app.RightPanel.Layout.Row = 1;
app.RightPanel.Layout.Column = 2;

% Create UIAxes
app.UIAxes = uiaxes(app.RightPanel);
title(app.UIAxes, 'Voltage')
xlabel(app.UIAxes, 'z [m]')
ylabel(app.UIAxes, 'V(z) [V]')
zlabel(app.UIAxes, 'Z')
app.UIAxes.FontSize = 12;
app.UIAxes.Position = [83 377 300 185];

% Create UIAxes2
app.UIAxes2 = uiaxes(app.RightPanel);
title(app.UIAxes2, 'Smith chart')
xlabel(app.UIAxes2, 'X')
ylabel(app.UIAxes2, 'Y')
zlabel(app.UIAxes2, 'Z')
app.UIAxes2.FontSize = 12;
app.UIAxes2.Position = [24 104 408 266];

% Create Output1EditFieldLabel
app.Output1EditFieldLabel = uilabel(app.
    RightPanel);
app.Output1EditFieldLabel.HorizontalAlignment
    = 'right';
app.Output1EditFieldLabel.Position = [45 74 52
     22];
app.Output1EditFieldLabel.Text = 'Output 1';

% Create Output1EditField
app.Output1EditField = uieditfield(app.
    RightPanel, 'text');
app.Output1EditField.Position = [110 76 323
    20];
```

```
            % Create Output2EditFieldLabel
            app.Output2EditFieldLabel = uilabel(app.
               RightPanel);
            app.Output2EditFieldLabel.HorizontalAlignment
               = 'right';
            app.Output2EditFieldLabel.Position = [45 44 52
                22];
            app.Output2EditFieldLabel.Text = 'Output 2';

            % Create Output2EditField
            app.Output2EditField = uieditfield(app.
               RightPanel, 'text');
            app.Output2EditField.Position = [110 46 323
               20];

            % Show the figure after all components are
               created
            app.UIFigure.Visible = 'on';
        end
    end

    % App creation and deletion
    methods (Access = public)

        % Construct app
        function app =
           transmission_Line_Solver13ProfCorrections

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.UIFigure)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)
```

```
            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```