Jakub Kopański
e-mail: J.Kopanski@imio.pw.edu.pl

Prowadzący:
prof. dr hab. Stanisław Rosłoniec

# Projektowanie układów mikrofalowych (PUM)
# Projekt

# Spis treści

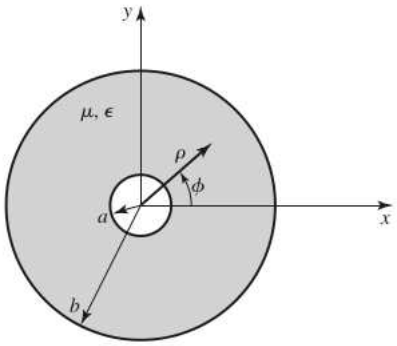
# Spis rysunków

# 1. Zadanie 1

## 1.1. Treść

Udowodnić, że powietrzna linia współosiowa o stosunku promieni przewodów zewnętrznego do wewnętrznego równym $\sqrt{e} = 1.648721271\ldots$ może przenosić falę elektromagnetyczną o największej mocy. Zadaną wielkością jest maksymalne natężenie pola elektrycznego, przy którym następuje przebicie elektryczne wypełniającego linię powietrza.

## 1.2. Rozwiązanie




Rysunek 1.1: Przekrój przez linie współosiową oraz symbole jej parametrów

Współosiową linie transmisyjną pokazano na rysunku 1.1. W treści zadania powiedziano, że mamy do czynienia z linią powietrzną więc $\epsilon = \epsilon_0$ oraz $\mu = \mu_0$. Moc fali przenoszonej przez linie jest równa:

$$P = \oint_S \bar{E}(x,y) \times \bar{H}(x,y)\ \mathrm{ds} \tag{1.1}$$

Ponieważ wyrażenia na pole elektryczne i magnetyczne w kartezjańskim układzie współrzędnych byłyby bardzo skomplikowane, należy zmienić układ współrzędnych na polarny:

$$P = \frac{1}{2}\int_0^{2\pi}\int_a^b \bar{E}(\rho,\phi)\cdot\bar{H}(\rho,\phi)\rho\ \mathrm{d}\rho\mathrm{d}\phi \tag{1.2}$$

Pole elektryczne i magnetyczne mają postać:

$$\bar{E}(\rho) = \frac{U_0\hat{\rho}}{\rho\ln\frac{b}{a}} \tag{1.3}$$

$$\bar{H}(\phi) = \frac{\bar{E}(\rho)\hat{\phi}}{\eta_0} \tag{1.4}$$

gdzie:
$\hat{\rho}$ - wersor w kierunku $\rho$,
$\hat{\phi}$ - wersor w kierunku $\phi$,
$\xi_0 = \sqrt{\frac{\mu_0}{\epsilon_0}}$ - impedancja falowa linii.

Możemy teraz policzyć moc fali przenoszonej przez linie:

$$\begin{aligned}
P &= \frac{1}{2}\int_0^{2\pi}\int_a^b \bar{E}(\rho,\phi)\cdot\bar{H}(\rho,\phi)\rho\ \mathrm{d}\rho\mathrm{d}\phi \\
&= \frac{1}{2}\int_0^{2\pi}\int_a^b \frac{1}{\xi_0}\frac{U_0^2}{\rho^2\ln^2\frac{b}{a}}\rho\ \mathrm{d}\rho\mathrm{d}\phi \\
&= \frac{1}{2}\int_0^{2\pi}\int_a^b \frac{1}{\xi_0}\frac{U_0^2}{\rho\ln^2\frac{b}{a}}\ \mathrm{d}\rho\mathrm{d}\phi \\
&= \frac{1}{2}\frac{1}{\xi_0}\frac{U_0^2}{\ln^2\frac{b}{a}}\int_0^{2\pi}\int_a^b \frac{1}{\rho}\ \mathrm{d}\rho\mathrm{d}\phi \\
&= \frac{1}{2}\frac{1}{\xi_0}\frac{U_0^2}{\ln^2\frac{b}{a}}\int_0^{2\pi}\ln|\rho|\Big|_a^b\ \mathrm{d}\phi \\
&= \frac{1}{2}\frac{1}{\xi_0}\frac{U_0^2}{\ln^2\frac{b}{a}}\ln\frac{b}{a}\int_0^{2\pi}1\ \mathrm{d}\phi \\
&= \frac{1}{2}\frac{1}{\xi_0}\frac{U_0^2}{\ln\frac{b}{a}}\phi\Big|_0^{2\pi} \\
&= \frac{2\pi}{\xi_0}\frac{U_0^2}{\ln\frac{b}{a}}
\end{aligned} \tag{1.5}$$

Korzystając z zależności:

$$U_0 = E_{max}\cdot a\ln\frac{b}{a} \tag{1.6}$$

można wyznaczyć moc fali propagującej się w linii w zależności od maksymalnego natężenia pola elektromagnetycznego ($E_{max}$). Podstawiając 1.6 do 1.5 otrzymuję się:

$$\begin{aligned}
P &= \frac{2\pi}{\xi_0}\frac{E_{max}^2\cdot a^2\ln^2\frac{b}{a}}{\ln\frac{b}{a}} \\
&= \frac{2\pi}{\sqrt{\frac{\mu_0}{\epsilon_0}}}E_{max}^2\cdot a^2\ln\frac{b}{a} \\
&= \underbrace{2\pi\sqrt{\frac{\epsilon_0}{\mu_0}}E_{max}^2}_{K}\cdot a^2\ln\frac{b}{a} \\
&= K\cdot a^2\ln\frac{b}{a}
\end{aligned} \tag{1.7}$$

$K$ we wzorze 1.7 jest stałe, zależne tylko od podanego w zadaniu maksymalnego natężenia pola.

Moc fali propagującej się w linii będzie maksymalna gdy pochodna mocy określonej wzorem 1.7 ($\frac{\mathrm{dP(a)}}{\mathrm{da}}$) będzie równa 0.

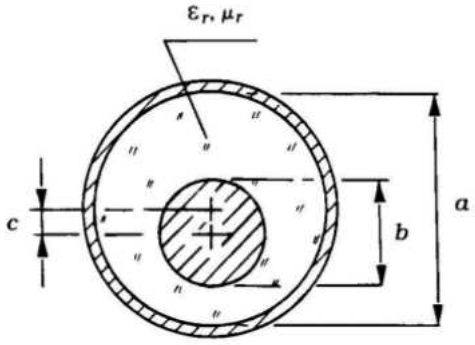
$$\begin{aligned}
\frac{\mathrm{dP(a)}}{\mathrm{da}} &= K\cdot 2a\ln\frac{b}{a} + K\cdot a^2(-\frac{1}{a}) \\
&= Ka[2\ln\frac{b}{a} - 1]
\end{aligned} \tag{1.8}$$

Wyrażenie 1.8 jest równe 0 gdy $2\ln\frac{b}{a} - 1 = 0$. Co z kolei przekłada się na warunek $\ln\frac{b}{a} = \frac{1}{2}$, który jest spełniony gdy $\frac{b}{a} = \sqrt{\mathrm{e}}$, co należało dowieść.

# 2. Zadanie 2

## 2.1. Treść

Dana jest powietrzna linia współosiowa o średnicach przewodów a = 7 $mm$ i b = 3.04 $mm$, patrz rys. 2.1. O ile należy przesunąć przewód wewnętrzny względem przewodu zewnętrznego (c =?) aby jej impedancja charakterystyczna zmieniła się o 5 Ω.




Rysunek 2.1: Linie z przesuniętym przewodem wewnętrznym

## 2.2. Rozwiązanie

Impedancja linii ekscentrycznej jest określona wzorem:

$$Z_0(x) = 59.952\sqrt{\frac{\mu_r}{\epsilon_r}}\ln\left(x+\sqrt{x^2-1}\right) \tag{2.1}$$

$$x = \frac{1}{2a}\left(b+\frac{a^2-4c^2}{b}\right) \tag{2.2}$$

Dla c = 0 mamy:

$$Z_0(x)\Big|_{c=0} = 59.952\sqrt{\frac{\mu_r}{\epsilon_r}}\ln\frac{a}{b} \tag{2.3}$$

Jest to zależność przybliżona. Dokładny wzór na impedancje linii współosiowej ma postać:

$$Z_0 = \frac{1}{2\pi}\sqrt{\frac{\mu}{\epsilon}}\ln\frac{a}{b} \tag{2.4}$$

Porównując podane wyżej zależności dla c = 0 mamy:
wzór dokładny : 50.0085378279
wzór przybliżony : 50.0031234918
co daję błąd równy 0.01%.

Zadanie można rozwiązać na 2 sposobu: analitycznie i numerycznie. kolejne zadania będzie można rozwiązać już tylko numerycznie więc porównując rozwiązanie analityczne i numeryczne zadania 2 można przetestować zaprogramowaną metodę newtona.

### 2.2.1. Rozwiązanie analityczne

Równanie z jakie należy rozwiązać to

$$\underbrace{59.952\sqrt{\frac{\mu_r}{\epsilon_r}}}_{k}\ln\left(x+\sqrt{x^2-1}\right)=\underbrace{\frac{1}{2\pi}\sqrt{\frac{\mu}{\epsilon}}\ln\frac{a}{b}-5}_{d} \tag{2.5}$$

$$\ln\left(x+\sqrt{x^2-1}\right)=\frac{d}{k} \tag{2.6}$$

Kluczowe jest spostrzeżenie, że $\operatorname{arch} x=\ln\left(x+\sqrt{x^2-1}\right)$. Biorąc cosinus hiperboliczny obu stron równania mamy

$$\operatorname{ch}\left(\operatorname{arch}\left(x\right)\right)=\operatorname{ch}\frac{d}{k} \tag{2.7}$$

$$x=\frac{1}{2}\left(e^{\frac{d}{k}}+e^{-\frac{d}{k}}\right) \tag{2.8}$$

Otrzymane $x$ należy podstawić do wzoru 2.2. Po kilku przekształceniach otrzymuję się zależność:

$$c=\sqrt{\left(a^2-ab2\operatorname{ch}\frac{d}{k}+b^2\right)\Big/4} \tag{2.9}$$

Podstawiając dane z treści zadania otrzymuję się przesunięcie $c=0.882307292061mm$.

### 2.2.2. Rozwiązanie numeryczne

W celu znalezienia wymaganego przesunięcia, należy znaleźć wartość x przy którym impedancja spełnia warunek:

$$Z_0(x)\Big|_{c=?}=Z_0(x)\Big|_{c=0}-5 \tag{2.10}$$

$$Z_0(x)\Big|_{c=?}-Z_0(x)\Big|_{c=0}+5=0 \tag{2.11}$$

a następnie znając $x$ wyznaczyć $c$. Oczywiście stosując rozwiązania numeryczne nie trzeba stosować się do kolejności wyznaczania (tzn. najpierw $x$ a potem $c$). Dysponując funkcjami wyznaczającymi wartość impedancji można odnaleźć od razu $c$.

Wynik rozwiązany za pomocą zaprogramowanego algorytmu Newtona-Raphsona wynosi:

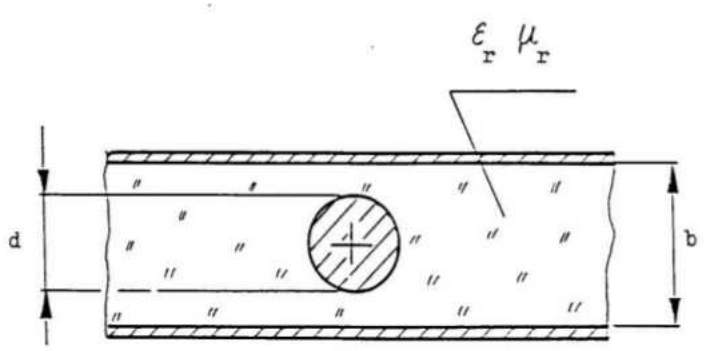
$$c_{numeryczne}=0.882307221883mm$$
$$c_{analityczne}=0.882307292061mm$$

różnica pomiędzy wynikami wynosi $8.12217580519e-11$, co jest zgodne z przyjętym kryterium zatrzymania pracy algorytmu na poziomie $1e-10$.

# 3. Zadanie 3

## 3.1. Treść

Zaprojektować powietrzną linię cylindryczno–płaską o przekroju poprzecznym jak na rys. 3.1 zakładając, że jej impedancja charakterystyczna jest równa $Z_0 = 30\ \Omega$. Odległość pomiędzy równoległymi przewodzącymi płaszczyznami tej linii jest równa $b = 9\ mm$. O ile zmieni się impedancja charakterystyczna tej linii (zaprojektowanej) po wypełnieniu jej bezstratnym dielektrykiem o $\epsilon_r = 2.04$ i $\mu_r = 1$.




Rysunek 3.1: Linia cylindryczno-płaska

## 3.2. Rozwiązanie

Impedancja charakterystyczna linii cylindryczno-płaskiej wyraża się wzorem:

$$Z_0\Big(\frac{d}{b}\Big) = 59.952\sqrt{\frac{\mu_r}{\epsilon_r}}\Big(\ln\frac{\sqrt{x}+\sqrt{y}}{\sqrt{x-y}} - \frac{R^4}{30} + 0.014R^8\Big), \tag{3.1}$$

gdzie:

$$R = \frac{\pi}{4}\frac{d}{b} \tag{3.2}$$

$$x = 1 + 2\,\mathrm{sh}^2\,(R) \tag{3.3}$$

$$y = 1 - 2\sin^2(R) \tag{3.4}$$

Zależność impedancji od wymiarów linii jest znacznie bardziej złożona niż w przypadku linii z zadania 2. Dlatego w tym przypadku nie można znaleźć rozwiązania w sposób analityczny. W celu określenia wymiarów linii należy rozwiązać równanie:

$$Z_0\Big(\frac{d}{b}\Big)\Big|_{d=9\ mm} - 30\Omega = 0 \tag{3.5}$$

Do znalezienia rozwiązania użyto zaprogramowanego poprzednio algorytmu Newtona-Raphsona.

Otrzymano następujące wyniki:

$$d = 6.73875214859\ mm \tag{3.6}$$

$$Z_0 = 30.0\ \Omega \tag{3.7}$$

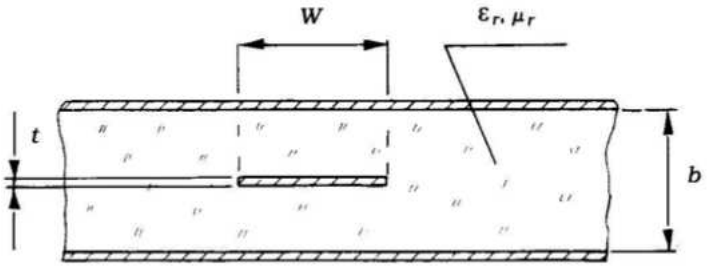
W celu odpowiedzi na drugie pytanie należy policzyć impedancje linii korzystając ze wzoru 3.1. Jednak zamiast $\mu_r = 1$ i $\epsilon_r = 1$, podstawić wartości określone w treści zadania. Uzyskana w ten sposób wartość impedancji wynosi $Z_0 = 21.0042012604\ \Omega$. Zmieni się ona zatem o $-8.99579873958\ \Omega$.

Zmiana impedancji wynika też bezpośrednio ze wzoru 3.1. Po wstawieniu dielektryka nowa wartość impedancji wyniesie $Z_0 \times \sqrt{\mu_r}$.

# 4. Zadanie 4

## 4.1. Treść

Zaprojektować symetryczną linię paskową, rys. 4.1, o impedancji charakterystycznej $Z_0 = 50\ \Omega$. Podłoże linii stanowi dielektryk o $\epsilon_r = 2.56$, $\mu_r = 1$ i grubości $b = 2.8\ mm$. Obliczenia wykonać, przy założeniu, że grubość przewodu wewnętrznego $t = 0\ mm$. Metodą różnic skończonych obliczyć impedancję charakterystyczną tej linii przyjmując, że przewód wewnętrzny $t = 0.150\ mm$.




Rysunek 4.1: Symetryczna linia paskowa

## 4.2. Rozwiązanie

### 4.2.1. Nieskończenie cienki przewód wewnętrzny

Impedancja charakterystyczna symetrycznej linii paskowej wyraża się wzorem:

$$Z_0(k) = 29.976\pi\sqrt{\frac{\mu_r}{\epsilon_r}}\frac{K(k)}{K'(k)} \tag{4.1}$$

gdzie:

$$k = \frac{1}{\mathrm{ch}\,(\frac{\pi w}{2b})} \tag{4.2}$$

przy założeniu nieskończenie cienkiego paska ($t = 0\ mm$). Z równania 4.2 można wyznaczyć szerokość paska:

$$w = \frac{2b}{\pi}\ln\big(\frac{1}{k} + \sqrt{\frac{1}{k^2} - 1}\big) \tag{4.3}$$

która to tworzy linie o impedanacji $Z_0$.

W pierwszym kroku z równania 4.1 można wyznaczyć stosunek całek eliptycznych $\frac{K(k)}{K'(k)}$:

$$\frac{K(k)}{K'(k)} = \frac{Z_0}{29.976\pi}\sqrt{\frac{\epsilon_r}{\mu_r}}. \tag{4.4}$$

Następnie można wyznaczyć stałą modularną $q$:

$$q = e^{-\pi\frac{K'(k)}{K(k)}}. \tag{4.5}$$

Stała modularna z równania 4.5 pozwala wyznaczyć wartość szeregów:

$$N = \sum_{i=1}^{\infty} q^{i\times(i-1)}, \tag{4.6}$$

$$D = 0.5 + \sum_{i=1}^{\infty} q^{i\times i}. \tag{4.7}$$

Szeregi 4.6 i 4.7 są szybko zbieżne i wystarczy już kilka pierwszych wyrazów aby uzyskać dobrą dokładność. W programie obliczanie wartości $N$ oraz $D$ zatrzymuję się automatycznie gdy osiągnięta została dokładność, która jest podawana jako parametr funkcji.

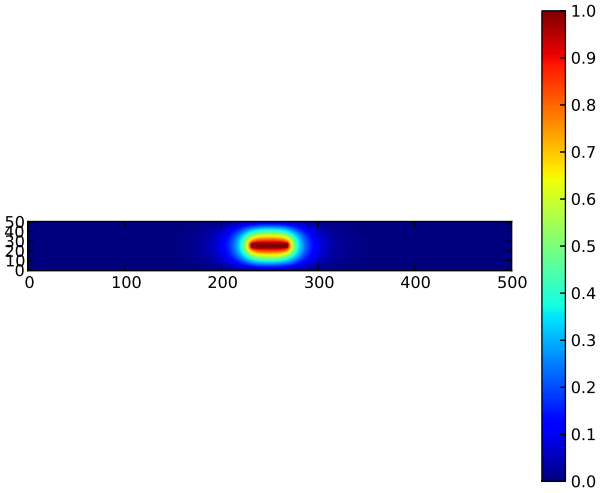
Korzystając z wyznaczonych wartości można obliczyć moduł $k$:

$$k = \sqrt{q}\Big(\frac{N}{D}\Big)^2. \tag{4.8}$$

Podstawiając wartość do równania 4.3 można obliczyć wymaganą szerokość paska. Dla danych podanych w treści zadania wymagana wartość $w = 2.06265925327\ mm$.

### 4.2.2. Przewód wewnętrzny o $t = 0.150\ mm$

W celu obliczenia dokładniejszej wartości impedancji skorzystano z metody różnic skończonych. W tym celu należy wyznaczyć rozkład potencjału w linii. Wykorzystano w tym celu algorytm Liebmanna opisany w [1]. Wynik został przedstawiony na rys. 4.2.




Rysunek 4.2: Rozkład potencjału w symetrycznej linii paskowej

Mając dany rozkład potencjału możemy obliczyć impedancję linii:

$$Z_0 = \sqrt{\frac{\mu}{\epsilon}}\frac{U}{\oint_{S_2} E_n \mathrm{ds}} \tag{4.9}$$

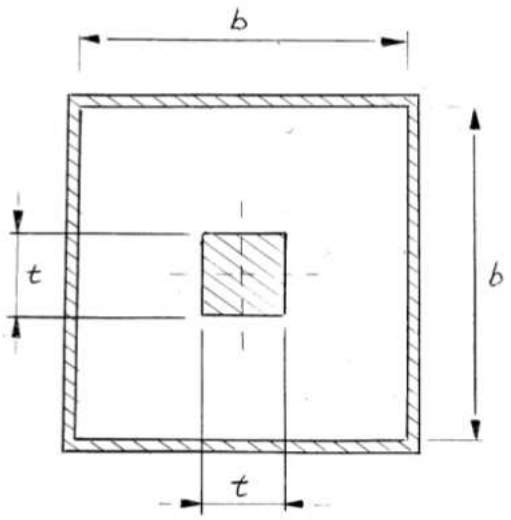
gdzie $U$ to napięcie pomiędzy przewodem wewnętrznym a zewnętrznym, $E_n$ to składowa normalna natężenia pola elektrycznego określona na linii brzegowej $S_2$ przewodu zewnętrznego.

W wyniku symulacji uzyskano impedancję równą $Z_0 = 45.2925852199\ \Omega$, dla przewodu o wymiarach określonych w sekcji 4.2.1. Daję to różnicę równą $4.70741478\ \Omega$ od wymaganych $50\ \Omega$.

# 5. Zadanie 5

## 5.1. Treść

Wykorzystując oprogramowanie napisane do rozwiązania zadania 4 wyznaczyć impedancję charakterystyczną powietrznej linii TEM o przekroju poprzecznym jak na rys. 5.1, przyjmując $b = 8\ mm$ i $t = 4\ mm$. Wynik otrzymany numerycznie porównać z wynikiem obliczonym według odpowiednich, przybliżonych wzorów wykorzystujących całki eliptyczne.

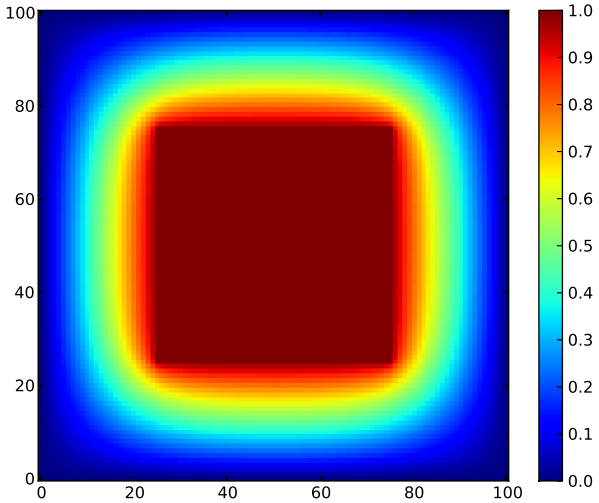



Rysunek 5.1: Linia rozważana w zadaniu 5

## 5.2. Rozwiązanie

Impedancja linii została wyznaczona w sposób opisany w sekcji 4.2.2. Dla linii określonej w niniejszym zadaniu wynosi ona $Z_0 = 36.7605605644\ \Omega$.

Impedancja obliczona według przybliżonego wzoru opartego o całki eliptyczne wynosi $Z_0 = 36.807171466\ \Omega$. Jest to wynik bardzo bliski uzyskanemu za pomocą metody różnic skończonych.

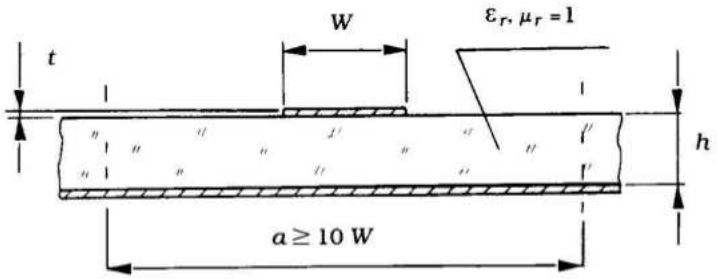



Rysunek 5.2: Rozkład potencjału w linii współosiowej z kwadratowymi przewodami

# 6. Zadanie 6

## 6.1. Treść

Zaprojektować niesymetryczną linię paskową, rys 6.1, o impedancji charakterystycznej $Z_0 = 50\ \Omega$. Podłoże linii stanowi dielektryk o $\epsilon_r = 2.56$, $\mu_r = 1$ i grubości $h = 1.4\ mm$. Obliczenia wykonać, przy założeniu, że grubość przewodu wewnętrznego $t = 0.0035\ mm$. Obliczyć długość fali w tak zaprojektowanej linii wiedząc, ze jej częstotliwość $f = 1.5\ GHz$.




Rysunek 6.1: Niesymetryczna linia paskowa

## 6.2. Rozwiązanie

### 6.2.1. Szerokość linii

Niesymetryczna linia paskowa, ze względu na niezwykle łatwe i tanie wytwarzanie, jest jedną z najpopularniejszych prowadnic falowych. Pomimo swojej popularności ciągle nie są znane analityczne zależności projektowe. Dlatego na potrzeby projektu posłużono się wzorami zawartymi w [2].

Rozwiązanie zadania polega na znalezieniu szerokości paska, jaki będzie tworzył linie o wymaganej impedancji. W tym celu należy numerycznie rozwiązać równanie:

$$Z_0(u, f) - Z_0 = 0 \tag{6.1}$$

gdzie:
$Z_0$ - wymagana impedancja
$u = \frac{W}{h}$ - stosunek od którego zależy impedancja
$f$ - częstotliwość pracy

Impedancje linii oblicza się wzorem:

$$Z_0(u, f) = \frac{60}{\sqrt{\epsilon_{ef}(f)}} \ln\left[\frac{f(u)}{u} + \sqrt{1 + \left(\frac{2}{u}\right)^2}\right] \tag{6.2}$$

Pomocnicze równania niezbędne do obliczenia impedancji zawarte są w [2].

Należy zwrócić uwagę na to, że wzór 6.2 jest słuszny w przypadku gdy przewód wewnętrzny jest nieskończenie cienki $t = 0$. Gdy chcemy uwzględnić grubość paska należy od $u$ dla którego równanie 6.1 jest spełnione odjąć poprawkę:

$$\Delta u = \frac{t}{2\pi h} \ln\left(1 + \frac{4eh}{t \operatorname{cth}^2 \sqrt{6.517u}}\right)\left(1 + \frac{1}{\operatorname{ch} \sqrt{\epsilon_r - 1}}\right) \tag{6.3}$$

Uwzględniając to wszystko zadanie rozwiązano korzystając z algorytmu Newtona-Raphsona napisanego dla poprzednich zadań. Wartość szerokości paska $w$ dla którego impedancja wynosi 50 Ω wynosi $w = 3.90022750273\ mm$.

### 6.2.2. Długość fali

Pole elektromagnetyczne w niesymetrycznej linii paskowej rozchodzi się, częściowo poprzez dielektryk a częściowo w powietrzu. Dlatego należy obliczyć efektywną przenikalność dielektryczną:

$$\epsilon_{eff}(u, f) = \frac{\epsilon_{eff}(u, 0) + \epsilon_r p(u, f)}{1 + p(u, f)} \tag{6.4}$$

$$\epsilon_{eff}(u, 0) = \frac{\epsilon_r + 1}{2} + \frac{\epsilon_r - 1}{2}\left(1 + \frac{10}{u}\right)^{-a(u) \times b(\epsilon_r)} \tag{6.5}$$

Mając obliczoną efektywną przenikalność elektryczną można obliczyć długość fali rozchodzącej się w linii.

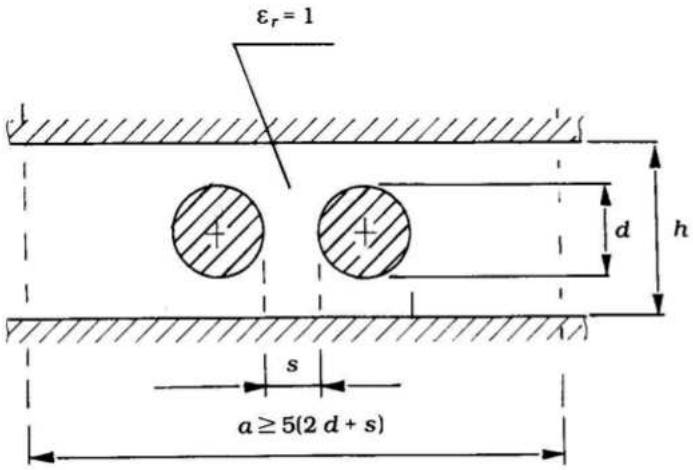
$$\lambda = \frac{c_{osr}}{f} = \frac{c}{\sqrt{\epsilon_{eff}} \times f} \tag{6.6}$$

Dla wartości określonych w treści zadania długość fali rozchodzącej się w zaprojektowanej linii wynosi: $\lambda = 13.6746810073\ cm$.

# 7. Zadanie 7

## 7.1. Treść

Zaprojektować powietrzne cylindryczno-płaskie linie sprzężone dla następujących danych: $Z_{0e} = 60\ \Omega$, $Z_{0o} = 40\ \Omega$. Obliczenia wykonać przy założeniu, że odległość pomiędzy dwoma zewnętrznymi płaszczyznami przewodzącymi jest równa $h = 8\ mm$, rys. 7.1.




Rysunek 7.1: Lnie cylindryczno-płaskie sprzężone

## 7.2. Rozwiązanie

Impedancje charakterystyczne powietrznych linii cylindryczno-płaskich, przy pobudzeniu synfazowym $Z_{0e}$ i przeciwfazowym $Z_{0o}$ określone są wzorami:

$$Z_{0e}(x,y) = 59.952 \ln\left(\frac{0.523962}{f_1(x) f_2(x,y) f_3(x,y)}\right) \tag{7.1}$$

$$Z_{0o}(x,y) = 59.952 \ln\left(\frac{0.523962 f_3(x,y)}{f_1(x) f_4(x,y)}\right) \tag{7.2}$$

gdzie:
$f_{(1/2/3/4)}$ - funkcje opisane w [2],
$x = \frac{d}{h}$ - stosunek średnicy przewodu do odstępu między płaszczyznami,
$y = \frac{s}{h}$ - stosunek odstępu między przewodami do odstępu między płaszczyznami.

Projektowanie linii sprowadza się do znalezienia takich $x$ i $y$ dla których spełnione są równania:

$$V1(x,y) = Z_{0e}(x,y) - Z_{0e} = 0 \tag{7.3}$$

$$V2(x,y) = Z_{0o}(x,y) - Z_{0o} = 0 \tag{7.4}$$

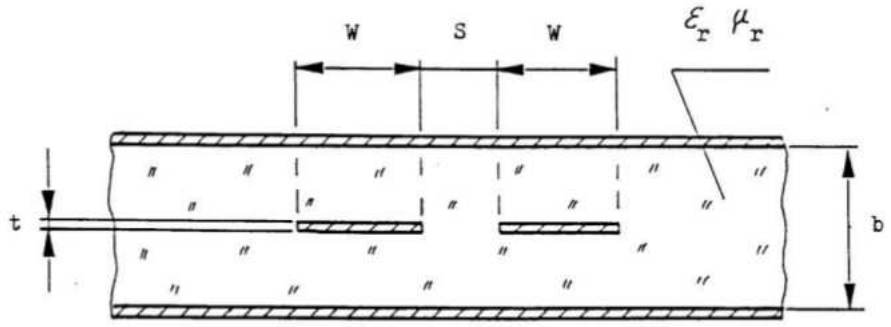
a finalnie $s$ i $d$.

Implementując metodę Newtona linia spełniająca wymagania postawione w treści zadania ma wymiary: $s = 1.97191812203\ mm$ i $d = 4.25688390818\ mm$.

# 8. Zadanie 8

## 8.1. Treść

Zaprojektować symetryczne linie paskowe sprzężone dla następujących danych o przekroju poprzecznym jak na rys. 8.1. Obliczenia wykonać dla $Z_{0e} = 60\ \Omega$, $Z_{0o} = 40\ \Omega$ przy założeniu, że podłoże linii stanowi dielektryk o $\epsilon_r = 2.56$, $\mu_r = 1$ i grubości $b = 2.8\ mm$. W trakcie obliczeń przyjąć, że grubość przewodów wewnętrznych $t \approx 0\ mm$.




Rysunek 8.1: Symetryczne linie paskowe

## 8.2. Rozwiązanie

Impedancje charakterystyczne symetrycznych sprzężonych linii paskowych wynoszą:

$$Z_{0e} = 29.976\pi\sqrt{\frac{\mu_r}{\epsilon_r}}\frac{K'(ke)}{K(ke)} \tag{8.1}$$

$$Z_{0o} = 29.976\pi\sqrt{\frac{\mu_r}{\epsilon_r}}\frac{K'(ko)}{K(ko)} \tag{8.2}$$

Powyższe wzory są słuszne gdy $t << b$, co jest spełnione dla $t \approx 0$ określonego w zadaniu.

W pierwszym kroku należy wyznaczyć:

$$\frac{K'(ke)}{K(ke)} = Z_{0e}/29.976\pi\sqrt{\frac{\mu_r}{\epsilon_r}} \tag{8.3}$$

$$\frac{K'(ko)}{K(ko)} = Z_{0o}/29.976\pi\sqrt{\frac{\mu_r}{\epsilon_r}} \tag{8.4}$$

Z ilorazu całek eliptycznych można wyznaczyć moduły $k_e$ i $k_o$. Następnie obliczamy parametry linii $W$ i $S$ zgodnie ze wzorami:

$$W = \frac{2b}{\pi}\,\mathrm{arth}(\sqrt{k_e k_o}) \tag{8.5}$$

$$S = \frac{2b}{\pi}\,\mathrm{arth}\left(\frac{k_e}{k_o}\right) - W \tag{8.6}$$

Dla wartości podanych w treści zadania potrzebne parametry linii to $w = 1.95755230148\ mm$ i $s = 0.384595243329\ mm$.

# 9. Zadanie 9

## 9.1. Treść

Zaprojektować tłumik rezystywny typu $T$ o tłumieniu $L = 10\ dB$, który włączony pomiędzy linie długie o impedancjach charakterystycznych $Z_{01} = 50\ \Omega$ i $Z_{02} = 60\ \Omega$ powinien zapewniać obustronne dopasowanie w nieskończenie szerokim paśmie częstotliwości. Zaprojektować równoważną wersję tego tłumika typu II.

## 9.2. Rozwiązanie

W pierwszym kroku należy sprawdzić realizowalność dzielnika. Należy wyznaczyć stosunek impedancji $r$:

$$r = \left(\frac{Z_{01}}{Z_{02}}\right)^{\pm 1} \tag{9.1}$$

przy czym znak przy wykładniku dobiera się tak, aby: $r > 1$.

Dla przypadku określonego w treści zadania:

$$\begin{aligned} r &= \frac{Z_{02}}{Z_{01}} = \frac{60}{50} \\ &= 1.2 \end{aligned}$$

Następnie można obliczyć minimalne tłumienie jakie wprowadza dzielnik:

$$L_{min} = 10 \log(\sqrt{r} + \sqrt{r-1}) \tag{9.2}$$

Podstawiając wartości określone w treści zadania otrzymuję się $L_{min} = 4.33507363245\ dB$ co jest mniejsze od wymaganego $L = 10\ dB$. Oznacza to, że tłumik jest realizowalny.

Projekt tłumików zaczyna się od przekształcenia wartości tłumienia z miary decybelowej na liniową:

$$N = 10^{(\frac{L}{10})} = 10 \tag{9.3}$$

### 9.2.1. Dzielnik typu $T$

W celu zaprojektowania tłumika typu $T$ wyznacza się wartości rezystancji zgodnie ze wzorami:

$$R_3 = \frac{2\sqrt{N \times Z_{01} \times Z_{02}}}{N-1} \qquad = 38.490017946\ \Omega \tag{9.4}$$

$$R_2 = Z_{02}\frac{N+1}{N-1} - R_3 \qquad = 34.8433153874\ \Omega \tag{9.5}$$

$$R_1 = Z_{01}\frac{N+1}{N-1} - R_3 \qquad = 22.6210931651\ \Omega \tag{9.6}$$

### 9.2.2. Dzielnik typu Π

W celu zaprojektowania tłumika typu Π wyznacza się wartości rezystancji zgodnie ze wzorami:

$$R_a = \frac{(N-1)\sqrt{Z_{01}Z_{02}}}{2\sqrt{N}} = 77.9422863406\ \Omega \tag{9.7}$$

$$R_b = \frac{Z_{01}R_a(N-1)}{R_a(N+1) - Z_{01}(N-1)} = 86.0997286466\ \Omega \tag{9.8}$$

$$R_c = \frac{Z_{02}R_a(N-1)}{R_a(N+1) - Z_{02}(N-1)} = 132.619585539\ \Omega \tag{9.9}$$

# 10. Zadanie 10

## 10.1. Treść

Zaprojektować schodkowy, ćwierćfalowy transformator impedancji o charakterystyce równomiernie falistej (Czebyszewa) dopasowujący dwie linie współosiowe o impedancjach charakterystycznych $Z_{01} = 30\ \Omega$ i $Z_{02} = 75\ \Omega$. Transformator ten powinien zapewniać w paśmie $2 \div 3\ GHz$ dopasowanie z $WFS \leq 1.12$. Projekt transformatora wykonać przy założeniu, że przewody zewnętrzne obu dopasowywanych linii mają średnicę $a = 7\ mm$. Zaprojektować równoważny wariant tego transformatora w postaci transformatora II klasy, tj. transformatora złożonego z niewspółmiernych odcinków linii o impedancjach charakterystycznych $Z_{01} = 30\ \Omega$ i $Z_{02} = 75\ \Omega$.

## 10.2. Rozwiązanie

### 10.2.1. Transformator schodkowy

Projekt transformatora rozpoczyna się od określenia ilości sekcji niezbędnych do realizacji. Minimalna ilość sekcji potrzebnych do realizacji transformatora jest większa lub równa:

$$\begin{aligned} n &\geq \frac{\operatorname{arch}\left(\frac{R-1}{\Gamma_d \times (r+1)}\right)}{\operatorname{arch}\left(\frac{1}{\cos(\pi \frac{1-\omega}{2})}\right)} = 1.47234760626 \\ n &= 2 \end{aligned} \tag{10.1}$$

gdzie:

$$\begin{aligned} R &= \frac{Z_{02}}{Z_{01}} && = 2.5, \\ \Gamma_d &= \frac{WFS - 1}{WFS + 1} && = 0.0566037735849. \end{aligned}$$

Następnie należy policzyć wartość impedancji kolejnych sekcji transformatora. Uzyskuję się je poprzez przemnożenie impedancji poprzedniego fragmentu linii poprzez współczynnik $V_K$. Sposób obliczania współczynników $V_K$ jest zależny od stopnia transformatora i dokładne wzory są podane w [2]. Dla przypadku podanego w treści zadania mamy:

$$\begin{aligned} Z_{01} &= 30\ \Omega \\ Z_1 &= Z_{01} \times V_1 && = 30\ \Omega \times 1.27247425628 = 38.1742276884\ \Omega \\ Z_2 &= Z_1 \times V_2 && = 38.1742276884\ \Omega \times 1.54398116863 = 58.9402886777\ \Omega \\ Z_{02} &= 75\ \Omega \end{aligned}$$

Znając impedancje kolejnych odcinków linii oraz transformatora możemy obliczyć szerokości przewodów wewnętrznych linii współosiowych realizujących dane impedancję. W tym celu należy posłużyć się zależnością:

$$b = a \times \exp\left(-\frac{Z_k}{59.952 \times \sqrt{\frac{\mu_r}{\epsilon_r}}}\right) \tag{10.2}$$

Dla danych z treści z zdania oraz obliczonych impedancji:

$$\begin{aligned} b_{01} &= 4.24401531258\ mm \\ b_1 &= 3.70307525226\ mm \\ b_2 &= 2.61898150779\ mm \\ b_{02} &= 2.00352744277\ mm \end{aligned}$$

Ostatnim etapem projektu jest wyznaczenie długości każdego z odcinków tworzących transformator. Długość elektryczna powinna wynosić $\frac{\lambda}{4}$. Długość fizyczną wyznacza się z zależności:

$$l\Big(\frac{\pi}{4}\Big) = \frac{c}{\sqrt{\mu_r/\epsilon_r}4f_0} \tag{10.3}$$

Dla danych z zadania wynosi: $l\Big(\frac{\pi}{4}\Big) = 2.99792458\ cm$.

### 10.2.2. Transformator impedancji II klasy

Transformator złożony z niewspółmiernych odcinków linii różni się od transformatora schodkowego tym, że składa się z odcinków linii o znanej impedancji charakterystycznej $Z_0$ i $R \times Z_0$, a projektowanie polega na doborze odpowiedniej ilości sekcji oraz długości elektrycznych odcinków. W przypadku tego zadania mamy $Z_0 = 30\ \Omega$, $R = \frac{75}{30} = 2.5$.

W pracy [2] przedstawiono metody projektowania dwu-, cztero-, sześcio- i ośmiosekcyjnych transformatorów impednacji II klasy. W przypadku tego zadania, ze względu warunek odpowiadania transformatorowi zaprojektowanemu w sekcji 10.2.1 należy zaprojektować transformator czterosekcyjny.

Długości elektryczne kolejnych sekcji transformatora wynoszą $\theta_i(f)$ i są związane z długością elektryczną pierwszej sekcji $\theta(f)$ zależnością:

$$\theta_i(f) = a_i\theta(f) \quad dla \quad i = 2, 3, \ldots n, \tag{10.4}$$

gdzie $a_i$ jest $i$-tą składową $n$-wymiarowego wektora $A = (1, a_2, a_3, \ldots, a_n)$

Projektowanie transformatora sprowadza się do aproksymacji funkcji wnoszonego tłumienia $L(A, \theta)$ w paśmie $[\theta_a, x\theta_a]$ gdzie $\theta_a$ i $x\theta_a$ oznaczają długości elektryczne pierwszej sekcji dla najmniejszej i największej częstotliwości pracy transformatora.

Z wymagań przedstawionych w [2] mamy: $a_2 = a_3$ i $a_4 = 1$. Dlatego szukane wartości to:

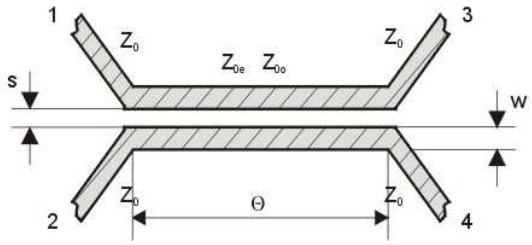
$$\theta_a = \theta(f_1) = \frac{V_4}{1+x} = 0.265397249812 \tag{10.5}$$

$$a_2 = \frac{f_3(r) + f_4(r)(2-x)}{V_4} = 2.73683917651 \tag{10.6}$$

# 11. Zadanie 11

## 11.1. Treść

Zaprojektować jednosekcyjny, zbliżeniowy sprzęgacz kierunkowy o sprzężeniu $C = 13\ dB$ przy częstotliwości $f = 1.34\ GHz$. Sprzęgacz zrealizować z odcinków symetrycznych linii paskowych (pojedynczych i sprzężonych) przyjmując, że podłoże linii stanowi dielektryk o $\epsilon_r = 2.56$, $\mu_r = 1$ i grubości $b = 2.8\ mm$. Projekt wykonać przy założeniu, że grubość przewodów wewnętrznych jest pomijalnie mała z grubością dielektryka $b = 2.8\ mm$ a impedancja charakterystyczna linii obciążających sprzęgacz jest równa $Z_0 = 50\ \Omega$.




Rysunek 11.1: Jednosekcyjny sprzęgacz zbliżeniowy

## 11.2. Rozwiązanie

Sprzęgacz z oznaczeniami przedstawiono na rys. 11.1. Rozważany sprzęgacz jest sprzęgaczem *w tył*, tzn. wrota sprzężone to wrota oznaczone numerem 2. Projekt sprzęgacza zaczyna się od wyznaczenia wartości napięciowego współczynnika sprzężenia:

$$k = 10^{-\frac{|C|}{20}} \tag{11.1}$$
$$= 0.223872113857$$

Następnie, na jego podstawie, wyznacza się wartości impedancji charakterystycznych:

$$Z_{0e} = Z_0\sqrt{\frac{1+k}{1-k}} \qquad = 62.7872381716\ \Omega \tag{11.2}$$

$$Z_{0o} = Z_0\sqrt{\frac{1-k}{1+k}} \qquad = 39.8170085642\ \Omega \tag{11.3}$$

$$\tag{11.4}$$

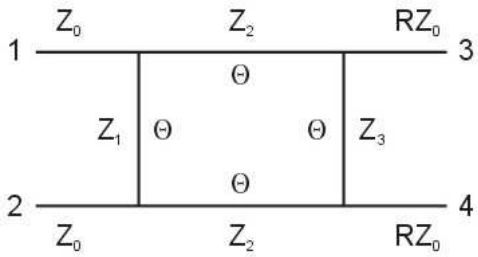
Wyznaczone impedancje to prawie koniec projektu. Realizacja sprzęgacza w technice linii paskowych sprowadza się do zagadnienia rozważanego w rozdziale 8. Szerokość i szczelina między ścieżkami opisane są zależnościami 8.5 i 8.6. Dla wartości podanych w treści zadania potrzebne parametry linii to $w = 1.85910996355\ mm$ i $s = 0.326090134191\ mm$.

Długość sprzęgacza powinna wynosić $\frac{1}{4} \times \lambda$. Dla zadanej częstotliwości i parametrów podłoża sprzęgacza $\lambda = 13.9828571828\ cm$, co daję długość sprzęgacza $l = 3.49571429571\ cm$.

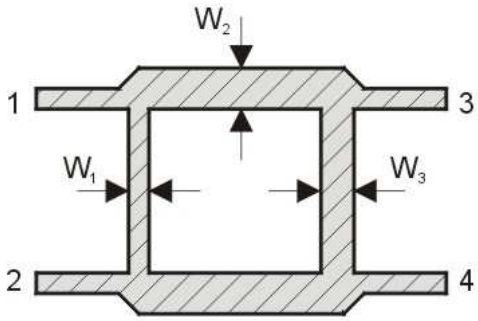
# 12. Zadanie 12

## 12.1. Treść

Zaprojektować dwugałęziowy sprzęgacz kierunkowy zapewniający przy częstotliwości $f = 1.34\ GHz$ sprzężenie $C = 3.9\ dB$. Sprzęgacz zrealizować z odcinków niesymetrycznej linii paskowej przyjmując, że podłoże linii stanowi dielektryk o $\epsilon_r = 4.34$, $\mu_r = 1$ i grubości $h = 1.4\ mm$. Projekt wykonać przy założeniu, że grubość przewodu wewnętrznego $t = 0.035\ mm$ a impedancja charakterystyczna linii obciążających sprzęgacz jest równa $Z_0 = 50\ \Omega$. Wyznaczyć częstotliwościową charakterystykę sprzężenia $C(f)\ [dB]$ w paśmie od $f = 1.75\ GHz$ do $f = 2.25\ GHz$.




Rysunek 12.1: Schemat elektryczny sprzęgacza




Rysunek 12.2: Realizacja sprzęgacza z niesymetrycznych linii paskowych

## 12.2. Rozwiązanie

### 12.2.1. Projekt sprzęgacza

Rysunki 12.1 i 12.2 przedstawiają projektowany sprzęgacz. Sygnał we wrotach 4 jest w przeciwfazie a we wrotach 3 w kwadraturze. Dwugałęziowy sprzęgacz kierunkowy może transformować impedancję.

Jednak w treści zadania zaznaczono, że jest on obciążony liniami o impedancji charakterystycznej $Z_0 = 50\ \Omega$. Stąd parametr $R = 1$.

Współczynnik napięciowego sprzężenia wynosi:

$$k = \frac{1}{\sqrt{10^{\frac{C}{10}} - 1}} = 0.829109611422 \tag{12.1}$$

Następnie podstawie współczynnika $k$, wyznacza się wartości impedancji charakterystycznych:

$$Z_1 = \frac{Z_0}{k} = 60.3056571908\ \Omega \tag{12.2}$$

$$Z_2 = Z_0\sqrt{\frac{R}{1+k^2}} = 38.4908989956\ \Omega \tag{12.3}$$

$$Z_3 = Z_0\frac{R}{k} = 60.3056571908\ \Omega \tag{12.4}$$

Wyznaczone impedancje należy zamienić na odcinki niesymetrycznych linii paskowych tak samo jak było to wykonane w rozdziale 6. Szerokości ścieżek wynoszą:

$$w1 = 1.90100127159\ mm$$
$$w2 = 4.01397627035\ mm$$
$$w3 = 1.90100127159\ mm$$

Podobnie jak w rozdziale 6 aby wyznaczyć długość odcinków ćwierćfalowych, należy obliczyć długość fali rozchodzącej się w linii. Długość fali zależy od efektywnej przenikalności dielektrycznej $\epsilon_{eff}$, która jest funkcją wymiarów oraz częstotliwości pracy. Dlatego obliczono 3 różne długości. Dla danych z treści zadania mamy:

$$\frac{\lambda_1}{4} = 3.11142895539\ cm$$
$$\frac{\lambda_2}{4} = 3.00913769452\ cm$$
$$\frac{\lambda_3}{4} = 3.11142895539\ cm$$

### 12.2.2. Charakterystyka sprzęgacza

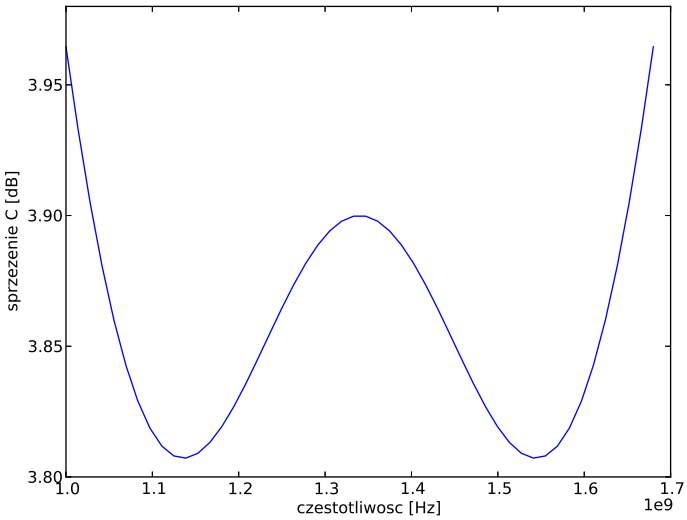
W celu wyznaczenia charakterystyki częstotliwościowej wartości sprzężenie należy posłużyć się zależnością:

$$C = 20\log\left(\frac{1}{|S_{14}|}\right) \tag{12.5}$$

$$S_{14} = \frac{1}{2}\left[\frac{1}{RD_e + jXD_e} - \frac{1}{RD_o + jXD_o}\right] \tag{12.6}$$

$$|S_{14}| = \frac{1}{2}\left[\frac{\sqrt{(RD_o - RD_e)^2 + (XD_o - XD_e)^2}}{(RD_e^2 + XD_e^2)(RD_o^2 + XD_e^2)}\right] \tag{12.7}$$

Dokładne zależności podane zostały w [2]. Charakterystykę sprzęgacza zaprezentowano na rys. 12.3.

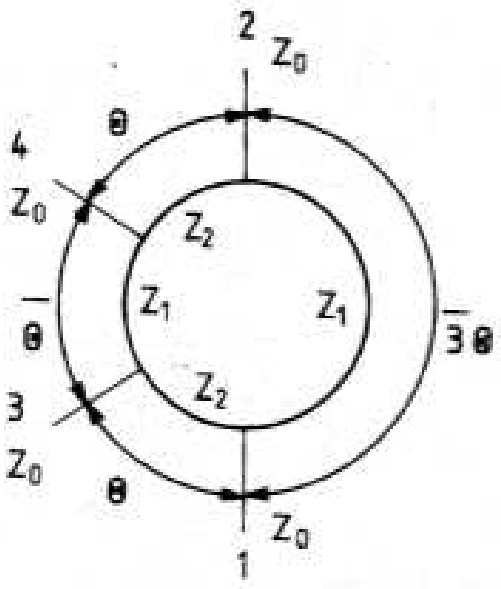



Rysunek 12.3: Charakterystyka częstotliwościowa sprzęgacza

# 13. Zadanie 13

## 13.1. Treść

Zaprojektować czteroramienny, pierścieniowy sprzęgacz kierunkowy zapewniający przy częstotliwości $f = 1.35\ GHz$ sprzężenie $C = 3.01\ dB$. Sprzęgacz zrealizować z odcinków niesymetrycznej linii paskowej przyjmując, że podłoże linii stanowi dielektryk o $\epsilon_r = 4.34$, $\mu_r = 1$ i grubości $h = 1.4\ mm$. Projekt wykonać przy założeniu, że grubość przewodu wewnętrznego $t = 0.035\ mm$ a impedancja charakterystyczna linii obciążających sprzęgacz jest równa $Z_0 = 50\ \Omega$. Wyznaczyć częstotliwościową charakterystykę sprzężenia $C(f)$ $[dB]$ w paśmie od $f = 1.25\ GHz$ do $f = 1.45\ GHz$.



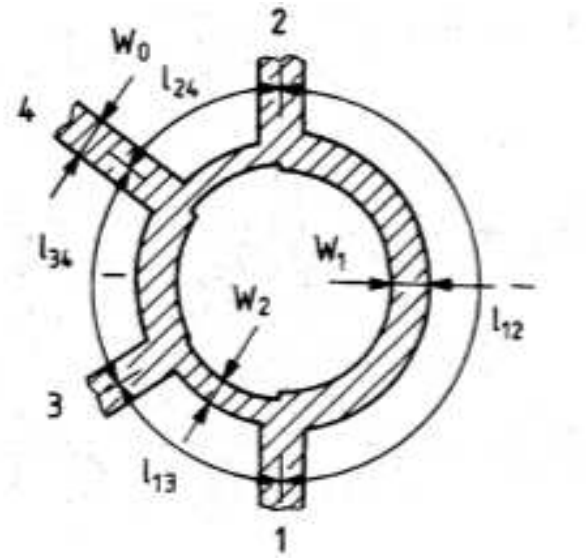

Rysunek 13.1: Schemat elektryczny sprzęgacza pierścieniowego

## 13.2. Rozwiązanie

### 13.2.1. Projekt sprzęgacza

Rysunki 13.1 i 13.2 przedstawiają projektowany sprzęgacz. Przy pobudzeniu wrót 4 wrota 2 i 3 są wyjściowe, synfazowe. Sprzężenie pomiędzy wrotami 3 i 4 wynosi:

$$C_{34} = 20 \log\left(\frac{1}{|S_{34}|}\right) = 10 \log\left(\frac{1}{y_1^2}\right) \tag{13.1}$$




Rysunek 13.2: Realizacja sprzęgacza z niesymetrycznych linii paskowych

gdzie:

$$y_1^2 + y_2^2 = 1 \tag{13.2}$$

$$y_1 = \frac{Z_0}{Z_1} \tag{13.3}$$

$$y_2 = \frac{Z_0}{Z_2} \tag{13.4}$$

jest warunkiem na idealne dopasowanie impedancyjne wrót sprzęgacza. Z równania 13.1 wynika zależność:

$$y_1 = \sqrt{10^{-\frac{C_{34}}{10}}} = 0.707131200681 \tag{13.5}$$

Co pozwala wyznaczyć kolejne wielkości:

$$y_2 = \sqrt{1 - y_1^2} \qquad = 0.707082360848 \tag{13.6}$$

$$Z_1 = \frac{Z_0}{y_1} \qquad = 70.7082362535\ \Omega \tag{13.7}$$

$$Z_2 = \frac{Z_0}{y_2} \qquad = 70.7131202368\ \Omega \tag{13.8}$$

$$\tag{13.9}$$

Wyznaczone impedancje należy zamienić na odcinki niesymetrycznych linii paskowych tak samo jak było to wykonane w rozdziale 6. Szerokości ścieżek wynoszą:

$$w_1 = 1.384465672\ mm$$
$$w_2 = 1.384261365\ mm$$

Podobnie jak w rozdziale 6 aby wyznaczyć długość odcinków ćwierćfalowych, należy obliczyć długość fali rozchodzącej się w linii. Długość fali zależy od efektywnej przenikalności dielektrycznej $\epsilon_{eff}$, która jest funkcją wymiarów oraz częstotliwości pracy. Dlatego obliczono 2 różne długości. Dla danych z treści zadania mamy:

$$\frac{\lambda_1}{4} = 3.1315202668\ cm$$
$$\frac{\lambda_2}{4} = 3.13153691782\ cm$$

### 13.2.2. Charakterystyka sprzęgacza

W celu wyznaczenia charakterystyki częstotliwościowej wartości sprzężenie należy posłużyć się zależnością:

$$C = 20 \log \left( \frac{1}{|S_{34}|} \right) \tag{13.10}$$

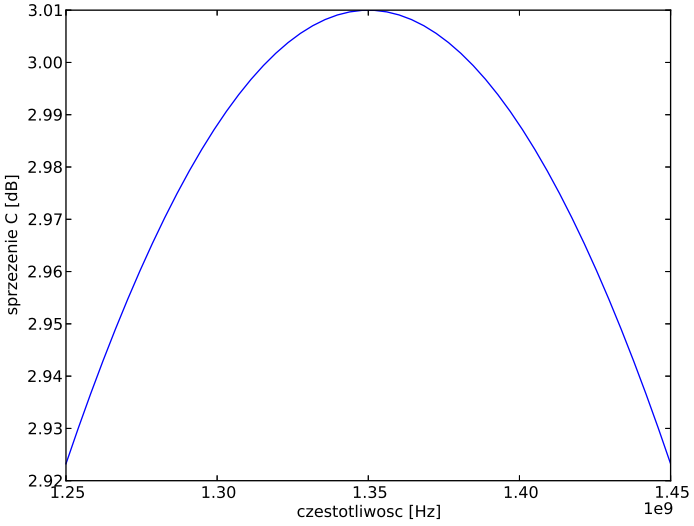
$$S_{34} = \frac{1}{2} \left( S_{22}^{++} - S_{22}^{+-} \right) \tag{13.11}$$

$$S_{22}^{++} = \frac{1 - A - B - jD}{1 + A + B + j(C + E)} \tag{13.12}$$

$$S_{22}^{+-} = \frac{1 - A' + B' - jD'}{1 + A' - B' - j(C' - E)} \tag{13.13}$$

$$\tag{13.14}$$

Dokładne zależności podane zostały w [2]. Charakterystykę sprzęgacza zaprezentowano na rys. 13.3.

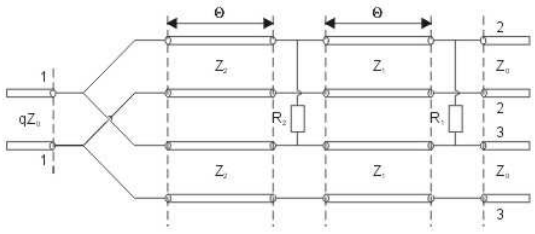



Rysunek 13.3: Charakterystyka częstotliwościowa sprzęgacza

# 14. Zadanie 14

## 14.1. Treść

Zaprojektować dwusekcyjny, trójwrotowy dzielnik sygnału mikrofalowego obciążony od strony wejścia rezystancją $Z_{01} = 35\ \Omega$. Wrota wyjściowe tego dzielnika są obciążone rezystancjami $Z_{02} = Z_{03} = 50\ \Omega$, odpowiednio. Projekt wykonać, przy założeniu, że środkowa częstotliwość pasma pracy $f_0 = 1.35\ GHz$. Wyznaczyć częstotliwościowe charakterystyki dopasowania we wrotach wejściowych $WFS(f)$ i izolacji (separacji) pomiędzy wrotami $I(f)[dB]$ w paśmie od $f_1 = 1.25\ GHz$ do $f_2 = 1.45\ GHz$.




Rysunek 14.1: Schemat elektryczny dzielnika

## 14.2. Rozwiązanie

### 14.2.1. Projekt dzielnika

Projektowany dzielnik jest przedstawiony na rys. 14.1. Ze względu na różne impedancje obciążenia i źródła dzielnik musi transformować impedancje. Przyjmując $Z_0 = 50\ \Omega = q \times Z_0$, stąd $q = 0.7$.

Impedancję sekcji dzielnika są opisane zależnościami:

$$Z_1 = Z_0 \times V_1 \tag{14.1}$$

$$Z_2 = Z_0 \times V_2 \tag{14.2}$$

gdzie:

$$V_1^2 = \sqrt{C^2 + 2q} + CV_2 \qquad = \frac{2q}{V_1^2} C = \frac{(2q-1)u_0^2}{2(2-u_0^2)} u_0 \qquad = \sin\left(\frac{w\pi}{4}\right) w = \frac{2(f_2 - f_1)}{f_1 + f_2} \tag{14.3}$$

Dla danych z treści zadania otrzymuje się:

$$Z_1 = 54.4190599649\ \Omega$$

$$Z_2 = 64.3157011947\ \Omega$$

Rezystory separujące oblicza się z zależności:

$$R_2 = \frac{2Z_1Z_2}{\sqrt{(Z_1+Z_2)(Z_2-Z_1\,\mathrm{ctg}^2(\theta_3))}} \tag{14.4}$$

$$R_1 = \frac{2R_2(Z_1+Z_2)}{R_2\frac{Z_1+Z_2}{Z_0}-2Z_2}\theta_3 \qquad = \frac{\pi}{2}\Big(1-\frac{2}{2\sqrt{2}}\Big) \tag{14.5}$$

co daje wynik:

$$R_2 = 80.3347745707\ \Omega$$
$$R_1 = 307.005233567\ \Omega$$

### 14.2.2. Charakterystyki dzielnika

W celu wyznaczenia charakterystyki częstotliwościowej dopasowania we wrotach wejściowych:

$$WFS(f) = \frac{1+|S_{11}|}{1-|S_{11}|} \tag{14.6}$$

Izolację z kolei określa zależność:

$$I(f) = 20\log\Big(\frac{1}{|S_{23}|}\Big) \tag{14.7}$$

Niezbędne parametry macierzy rozproszenia S:

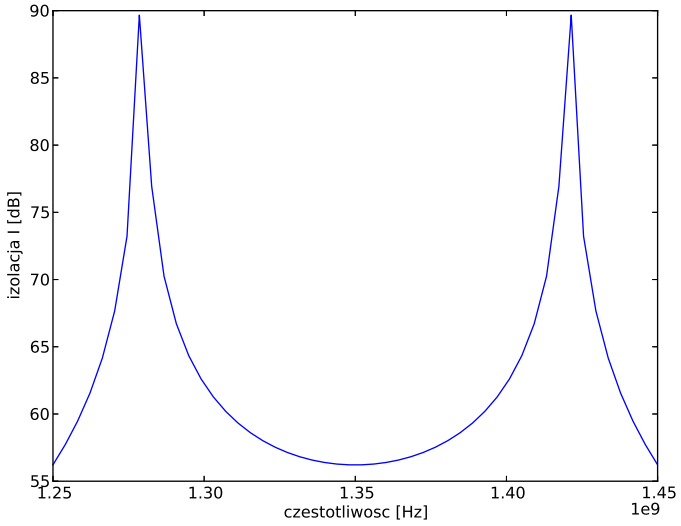
$$S_{11} = \frac{-RN+jXN}{RD+jXD} \tag{14.8}$$

$$S_{23} = \frac{1}{2}\left[\frac{RN+jXN}{RD+jXD} - S_{22}^{+-}\right] \tag{14.9}$$

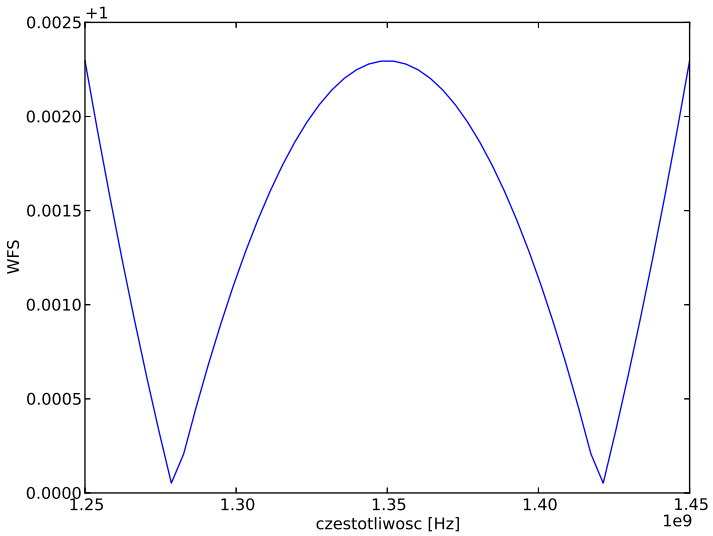
$$S_{22}^{+-} = \frac{A+jB}{C+jD} \tag{14.10}$$

$$\tag{14.11}$$

Dokładne zależności podane zostały w [2]. Charakterystyki dzielnika zaprezentowano na rys. 14.2 i 14.3.




Rysunek 14.2: Charakterystyka izolacji dzielnika

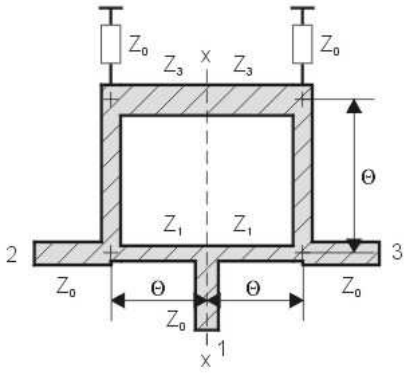



Rysunek 14.3: WFS na wejściu dzielnika

# 15. Zadanie 15

## 15.1. Treść

Zaprojektować dzielnik sygnału mikrofalowego typu Gysel'a przyjmując $f_0 = 1.35\ GHz$ i $Z_0 = 50\ \Omega$. Projekt dzielnika wykonać przy założeniu, że jest on realizowany z odcinków powietrznej, symetrycznej linii paskowej o grubości $b = 8\ mm$. Grubość przewodu wewnętrznego $t = 0.8\ mm$. Obliczyć charakterystykę sprzężenia $C(f)[dB]$ w paśmie od $f_1 = 1.25\ GHz$ do $f_2 = 1.45\ GHz$.




Rysunek 15.1: Zarys konstrukcyjny projektowanego dzielnika

## 15.2. Rozwiązanie

### 15.2.1. Projekt dzielnika

Projektowany dzielnik jest przedstawiony na rys. 15.1. Impedancje poszczególnych sekcji dzielnika opisane są zależnościami:

$$Z_0 = 50\ \Omega$$

$$Z_1 = \sqrt{2} * Z_0 \qquad = 70.7106781187\ \Omega \tag{15.1}$$

$$Z_3 = \frac{Z_0}{\sqrt{2}} \qquad = 35.3553390593\ \Omega \tag{15.2}$$

$$\tag{15.3}$$

W celu wyznaczenia parametrów realizacji dzielnika za pomocą symetrycznych linii paskowych wykorzystano rozwiązanie zadania 4. Otrzymane szerokości przewodów wewnętrznych wynoszą:

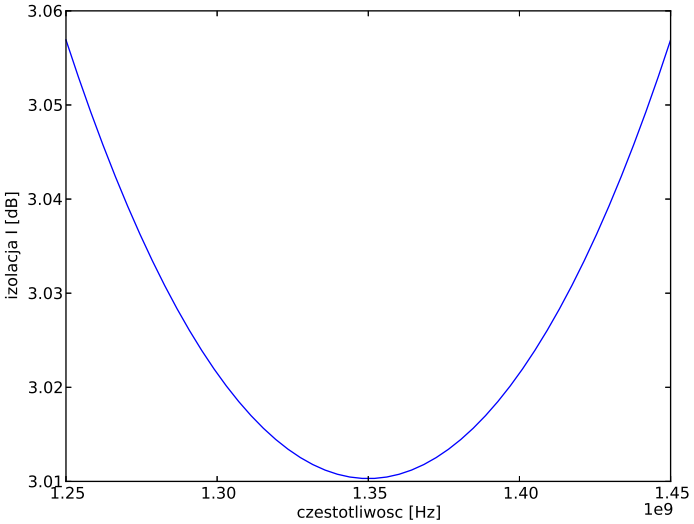
$$w_0 = 8.58657086038\ mm$$
$$w_1 = 4.65264251488\ mm$$
$$w_3 = 14.1720836113\ mm$$

### 15.2.2. Charakterystyka sprzężenia dzielnika

W celu wyznaczenia częstotliwościowej charakterystyki sprzężenia:

$$C(f) = 20 \log \frac{1}{|S_{12}|} \tag{15.4}$$

$$S_{12} = \frac{2}{R_{22} + jX_{22}} \tag{15.5}$$



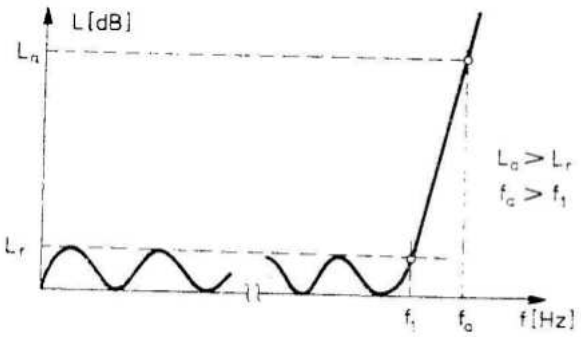

Rysunek 15.2: Charakterystyka sprzężenia dzielnika

Dokładne zależności podane zostały w [2]. Charakterystyki dzielnika zaprezentowano na rys. 15.2.

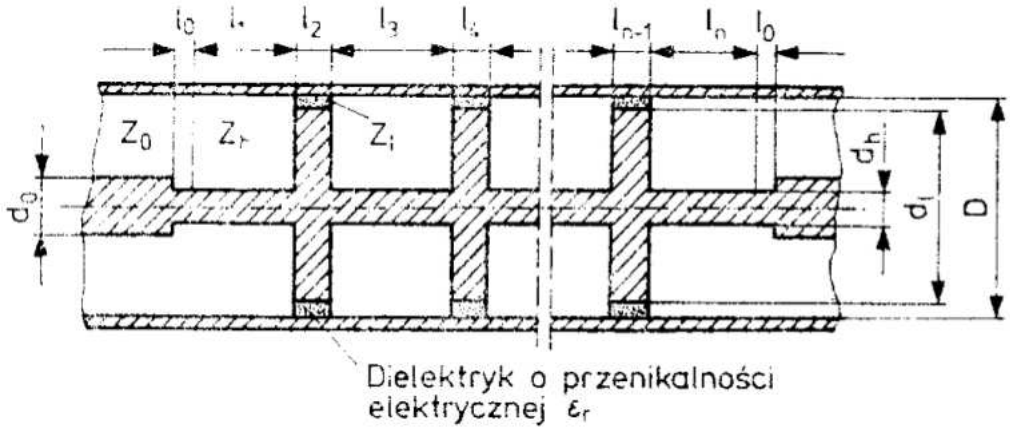
# 16. Zadanie 16

## 16.1. Treść

Zaprojektować filtr dolnoprzepustowy (FDP) o charakterystyce równomiernie falistej, rys. 16.1, dla następujących danych: $Z_0 = 50\ \Omega$, $f_1 = 10^9\ Hz$, $L_r = 0.2\ dB$, $f_a = 1.43 \times 10^9\ Hz$ i $L_a = 30\ dB$. Filtr zrealizować z odcinków linii współosiowej o średnicy przewodu zewnętrznego $D = 7\ mm$, rys.16.2. Obliczenia wykonać przy założeniu, że impedancje charakterystyczne niskoomowych i wysokoomowych sekcji filtru są równe odpowiednio $Z_l = 10\ \Omega$ i $Z_h = 120\ \Omega$. Ponadto założyć, że niskoomowe sekcje filtru są odcinkami linii współosiowej wypełnionej dielektrykiem o $\epsilon_r = 2.05$ i $\mu_r = 1$.




Rysunek 16.1: Charakterystyka projektowanego filtru




Rysunek 16.2: Realizacja filtru przy użyciu linii współosiowej

Tabela 16.1: Parametry zaprojektowanego filtru

| i | g | L [nH] | C [pF] | l [mm] | $d_l$ [mm] | $d_h$ [mm] |
|---|---|---|---|---|---|---|
| 0 | 1.0 | | | | 3.04015838775 | 3.04015838775 |
| 1 | 1.37229535453 | 10.9203794528 | — | 28.6038828188 | — | 0.945831227808 |
| 2 | 1.37819320784 | — | 4.38692523125 | 6.05098425027 | 5.51288864551 | |
| 3 | 2.27568854642 | 18.109354055 | — | 57.3968306129 | — | 0.945831227808 |
| 4 | 1.50014664009 | — | 4.77511506265 | 6.02588104449 | 5.51288864551 | |
| 5 | 2.27568854642 | 18.109354055 | — | 57.3968306129 | — | 0.945831227808 |
| 6 | 1.37819320784 | — | 4.38692523125 | 6.05098425027 | 5.51288864551 | |
| 7 | 1.37229535453 | 10.9203794528 | — | 28.6038828188 | — | 0.945831227808 |
| 8 | 1.07602182984 | | | | | |

## 16.2. Rozwiązanie

W pierwszym kroku należy obliczyć minimalną ilość sekcji filtru:

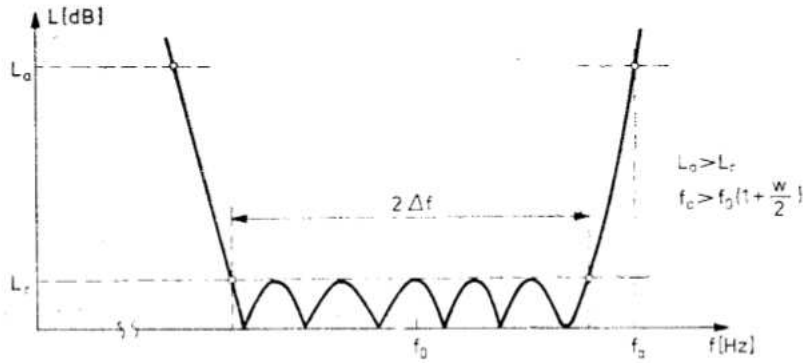
$$n \geq \frac{arch\sqrt{\frac{L'_a-1}{L'_r-1}}}{arch\left(\frac{f_a}{f_1}\right)} \tag{16.1}$$
$$= 7$$

Następnie należy obliczyć parametry filtru dolnoprzepustowego a na ich podstawie wartości elementów filtru o parametrach skupionych. Wyniki tych obliczeń przedstawia tabela 16.1. Długość $l_o$ stanowiąca poprawkę uwzględniającą pojemność $C_{f0}$ wynosi $l_o = 2.60235408442\ mm$.

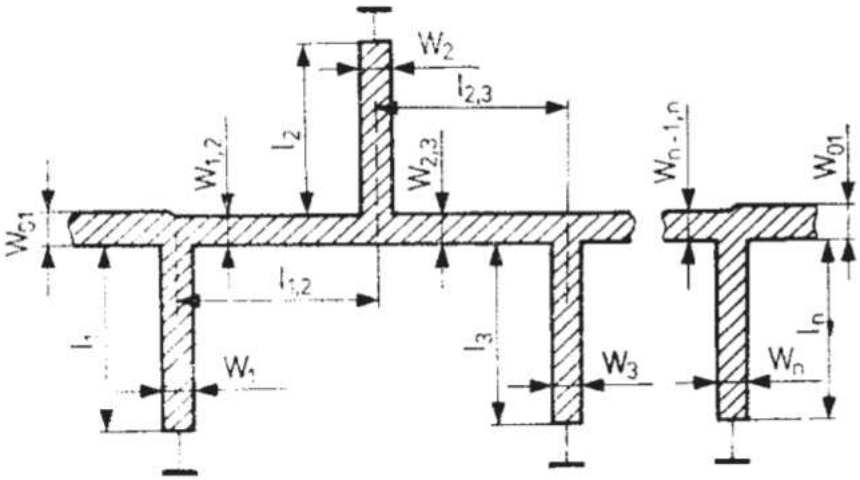
# 17. Zadanie 17

## 17.1. Treść

Zaprojektować czebyszewowski filtr pasmowo przepustowy (FPP, rys. 17.1) o strukturze paskowej jak na rys. 17.2 dla następujących danych: $Z_0 = 50\ \Omega$, $f_0 = 2.8 \times 10^9\ Hz$, $w = 0.1$, $L_r = 0.2\ dB$, $f_a = 3.2 \times 10^9\ Hz$ i $L_a = 30\ dB$. Filtr zrealizować z odcinków symetrycznej linii paskowej opisanej w zadaniu 4.




Rysunek 17.1: Charakterystyka projektowanego filtru




Rysunek 17.2: Realizacja filtru przy użyciu symetrycznej linii paskowej

Tabela 17.1: Parametry zaprojektowanego filtru

| i | g | $Z_0$ [Ω] | $w$ [mm] | $l$ [mm] |
|---|---|---|---|---|
| 0 | 1.0 | | | |
| 1 | 1.30287657175 | 3.20338115776 | 47.5830406797 | 16.7294898437 |
| 2 | 1.28442456136 | 3.46442342228 | 43.8597176927 | 16.7294898437 |
| 3 | 1.97619882964 | 3.39627708893 | 44.7764554109 | 16.7294898437 |
| 4 | 0.84680075915 | 4.9956490685 | 29.8615873107 | 16.7294898437 |
| 5 | 1.0 | | | |

## 17.2. Rozwiązanie

W pierwszym kroku należy obliczyć minimalną ilość sekcji filtru:

$$n \geq \frac{arch\sqrt{\frac{L'_a-1}{L'_r-1}}}{arch\left(\frac{\sin\frac{\pi w_a}{4}}{\sin\frac{\pi w}{4}}\right)} \tag{17.1}$$
$$= 4$$

Następnie należy obliczyć parametry filtru dolnoprzepustowego a na ich podstawie wartości elementów filtru o parametrach skupionych. Wyniki tych obliczeń przedstawia tabela 17.1.

# A. Kody użytych funkcji

## A.1. pum/lines.py

```
1  import scipy.constants as const
2  import numpy as np
3  import algorithms as alg
4
5  def coax_z(a, b, mu, epsilon): # impedance of coaxial line
6      return np.sqrt(const.mu_0 * mu / ( const.epsilon_0 * epsilon)) * np.log( a / b) / ( 2 * const.pi)
7
8  def skew_coax_z( c, a, b, mu, epsilon):
9      x = ( b + ( a * a - ( 4 * c * c)) / b) / (2 * a)
10     return 59.952 * np.sqrt( mu / epsilon) * np.log( x + np.sqrt( x * x - 1))
11
12 def square_coax( b, t, mu, epsilon):
13     kkp = ( 1 - ( t / b)) / ( 1 + ( t / b))
14     qp = np.exp( - const.pi * ( 1 / kkp))
15     p = np.sqrt( qp) * ( ( alg.n_fun( qp) / alg.d_fun( qp)) ** 2)
16     q = np.sqrt( 1 - ( p ** 2))
17     k = ( ( p - q) ** 2) / ( ( p + q) ** 2)
18     kpk = 1 / alg.k_int( k)
19     return 47.086 * kpk / np.sqrt( epsilon)
20
21 def cylindrical_flat( d, b, mu, epsilon): # impedance of cylindrical flat line
22     R = const.pi * d / (4 * b)
23     x = 1 + ( 2 * ( np.sinh( R) ** 2))
24     y = 1 - ( 2 * ( np.sin( R) ** 2))
25     return 59.952 * np.sqrt( mu / epsilon) * \
26         ( np.log( ( np.sqrt( x) + np.sqrt( y)) / \
27                   ( np.sqrt( x - y))) - \
28                   ( R**4 / 30) + \
29                   ( 0.014 * ( R**8)))
30
31 def stripline(w, t, h, mu, epsilon):
32     m  = 6 * ( h - t) / ( 3 * h - t)
33     dw = t * ( 1 - \
34                ( np.log( ( ( t / ( 2 * h - t)) ** 2) + \
35                          ( ( ( 0.0796 * t) / ( 1.1 * t + W)) ** m)) / 2)) / const.pi
36     A  = 4 * ( h - t) / ( const.pi * ( w + dw))
37     return 30 * np.log( 1 + A * ( 2 * A + np.sqrt( 4 * ( A ** 2) + 6.27))) / np.sqrt( epsilon)
38
39 def microstrip(w, t, h, f, mu, epsilon):
40     u  = w / h
41     if t != 0:
42         du = ( t / ( 2 * const.pi * h)) * np.log( 1 + ( ( 4 * np.exp(1) * h) / ( t * ( ( 1 / np.tanh( np.sqrt(
43         u += du
44     a  = 1 + ( ( 1 / 49) * np.log( ( ( u**4) + ( ( u / 52)**2)) / ( ( u**4) + 0.432))) + ( ( 1 / 18.7) * np.lo
45     b  = 0.564 * ( ( ( epsilon - 0.9) / ( epsilon + 3))**0.053)
46     c  = 0.33622 * ( 1 - np.exp( -0.03442 * epsilon))
47     d  = 3.751 - ( 2.75 * np.exp( - ( ( epsilon / 15.916)**8)))
48     fn = f * h * ( 10**-7)
49     g  = 1 - np.exp( - ( ( fn / 3.87)**4.97))
50     k  = 0.363 * g * np.exp( -4.6 * u)
51     m  = 0.525 / ( ( 1 + ( 0.157 * fn))**20)
52     n  = 0.27488 + ( 0.6315 * u) + ( m * u) - ( 0.065683 * np.exp( -8.7513 * u))
53     p  = n * c * ( ( ( 1.844 * fn) + ( k * d * fn))**1.5763)
54     eps_0 = ( ( epsilon + 1) / 2) + ( ( ( epsilon - 1) / 2) * ( ( 1 + ( 10 / u))**( -a * b)))
55     eps_eff = ( eps_0 + ( epsilon * p)) / ( 1 + p)
56     f = 6 + ( ( 2 * const.pi - 6) * np.exp( - ( ( 30.666 / u)**0.7528)))
57     return ( 60 / np.sqrt( eps_eff)) * np.log( ( f / u) + np.sqrt( 1 + ( 2 / u)**2))
58
59 def cylindrical_flat_coupled( s, d, h, mu, epsilon):
60     x = d / h
61     y = s / h
62     a = 1 + np.exp( 16 * x - 18.272)
63     b = np.sqrt( 5.905 - ( x**4))
64     c = ( -0.8107 * ( y**3)) + \
65         ( 1.3401 * ( y**2)) - \
66         ( 0.6929 * y) + \
67         ( 1.0892) + \
68         ( 0.014002 / y) - \
```

```
        ( 0.000636 / ( y**2))
    d = 0.11 - ( 0.83 * y) + ( 1.64 * ( y**2)) - ( y**3)
    e = -0.15 * np.exp( -13 * x)
    g = 2.23 * np.exp( ( -7.01 * y) + \
                       ( 10.24 * ( y**2)) - \
                       ( 27.58 * ( y**3)))
    k = 1 + ( 0.01 * ( ( -0.0726) - \
                       ( 0.2145 / y) + \
                       ( 0.222573 / ( y ** 2)) - \
                       ( 0.012823 / ( y ** 3))))
    l = 0.01 * ( ( -0.26) + \
                 ( 0.6866 / y) + \
                 ( 0.0831 / ( y ** 2)) - \
                 ( 0.0076 / ( y ** 3)))
    m = ( -0.1098) + \
        ( 1.2138 * x) - \
        ( 2.2535 * ( x ** 2)) + \
        ( 1.1313 * ( x ** 3))
    n = ( -0.019) - \
        ( 0.016   / y) + \
        ( 0.0362  / ( y ** 2)) - \
        ( 0.00234 / ( y ** 3))
    f1 = x * a / b
    if y < 0.9:
        f2 = c - ( x * d) + ( e * g)
    else:
        f2 = 1 + ( 0.004 * np.exp( 0.9 - y))
    f3 = np.tanh( const.pi * ( x + y) / 2)
    if y < 0.9:
        f4 = k - ( x * l) + ( m * n)
    else:
        f4 = 1
    Z0e = 59.952 * np.log( 0.523962 / ( f1 * f2 * f3))
    Z0o = 59.952 * np.log( ( 0.523962 * f3) / ( f1 * f4))
    return ( Z0e, Z0o)

def stripline_coupled( w, s, b, t, mu, epsilon):
    if t == 0:
        ke = np.tanh( ( const.pi * w) / ( 2 * b)) * \
             np.tanh( ( const.pi / 2) * ( ( w + s) / b))
        ko = np.tanh( ( const.pi * w) / ( 2 * b)) * \
             ( 1 / np.tanh( ( const.pi / 2) * ( ( w + s) / b)))
        kpke = 1 / alg.k_int( ke)
        kpko = 1 / alg.k_int( ko)
        Z0e = 29.976 * const.pi * np.sqrt( mu / epsilon) * kpke
        Z0o = 29.976 * const.pi * np.sqrt( mu / epsilon) * kpko
    else:
        theta = ( const.pi * s) / ( 2 * b)
        ae  = ( np.log( 2) + np.log( 1 + np.tanh( theta))) / \
              ( 2 * const.pi * np.log( 2))
        ao  = ( np.log( 2) + np.log( 1 + ( 1 / np.tanh( theta)))) / \
              ( 2 * const.pi * np.log( 2))
        c   = 2 * np.log( ( 2 * b - t) / ( b - t)) - \
              ( ( t / b) * np.log( ( t * ( 2 * b - t)) / ( ( b - t) ** 2)))
        Z0e = ( 30 * const.pi * ( b - t)) / \
              ( np.sqrt( epsilon) * ( w + ( ae * b * c)))
        Z0o = ( 30 * const.pi * ( b - t)) / \
              ( np.sqrt( epsilon) * ( w + ( ao * b * c)))
    return (Z0e, Z0o)
```

## A.2. pum/fdm.py

```
import scipy.constants as const
import scipy.integrate as integ
import numpy as np
import matplotlib.pyplot as plt
import shapely.geometry as shp

class pt:
    """Single point of finite difference method net"""
    def __init__( self, u = 0.0, p = 1.0, q = 1.0, r = 1.0):
        self.__dict__['u'] = float( u)
        self.p = float( p)
        self.q = float( q)
        self.r = float( r)

class metal( pt):
    """Metal point.

```

```
18      Assign new potential only upon creation"""
19      def __setattr__( self, name, val):
20          if name == 'u':
21              pass
22          else:
23              self.__dict__[name] = val
24
25  class imaginary( pt):
26      """Imaginary point.
27
28      Used when dividing structure"""
29      def __init__( self, pt):
30          self.ref = pt
31
32      def __getattr__( self, name):
33          if name == 'u':
34              return self.ref.u
35
36  class struct:
37      """Microwave structure to analyze with finite difference method"""
38      def __init__( self, a, b, I, J, mu=1, eps=1):
39          self.a = a
40          self.b = b
41          self.I = I
42          self.J = J
43          self.mu = mu
44          self.eps = eps
45          self.xstep = a / I
46          self.ystep = b / J
47          self.st = [[pt(0.0) for x in xrange(self.I+1)] \
48                     for z in range(self.J+1)]
49          self.bl = []
50
51      def add_rect( self, a, b, x, y, u):
52          xs = a / 2
53          ys = b / 2
54          self.bl.append( ( shp.Polygon( [ (x - xs, y + ys), \
55                                           (x + xs, y + ys), \
56                                           (x + xs, y - ys), \
57                                           (x - xs, y - ys)]), u))
58
59      def add_plane( self, sx, sy, ex, ey, u):
60          self.bl.append( ( shp.LineString( [(sx, sy), (ex, ey)]), u))
61
62      def init( self):
63          for j in range( self.J + 1):
64              for i in range( self.I + 1):
65                  # is it metal?
66                  for item in self.bl:
67                      #if item[0].contains( shp.Point( i * self.xstep, j * self.ystep)):
68                      if not item[0].disjoint( shp.Point( i * self.xstep, j * self.ystep)):
69                          self.st[j][i] = metal( item[1])
70                          continue
71                  if not isinstance( self.st[j][i], metal):
72                      # if not is is standard point
73                      # but look for metal vicinity
74                      for item in self.bl:
75                          p = self.ystep / self.xstep
76                          q = 1.0
77                          r = p
78                          step = self.xstep
79                          #if item[0].contains( shp.Point( (i - 1) * self.xstep, (j + 0) * self.ystep)):
80                          if not item[0].disjoint( shp.Point( (i - 1) * self.xstep, (j + 0) * self.ystep)):
81                              # calculate new p, q, r values
82                              line = shp.LineString( [( i * self.xstep, j * self.ystep), ( (i - 1) * self.xstep,
83                              inter = item[0].intersection( line)
84                              step = shp.Point( i * self.xstep, j * self.ystep).distance( inter)
85                              p = p * self.xstep / step
86                              q = q * self.xstep / step
87                              r = r * self.xstep / step
88                          #if item[0].contains( shp.Point( (i + 1) * self.xstep, (j + 0) * self.ystep)):
89                          if not item[0].disjoint( shp.Point( (i + 1) * self.xstep, (j + 0) * self.ystep)):
90                              # calculate new q value
91                              line = shp.LineString( [( i * self.xstep, j * self.ystep), ( (i + 1) * self.xstep,
92                              inter = item[0].intersection( line)
93                              #q = ( inter.bounds[0] - ( i * self.xstep)) / step
94                              q = shp.Point( i * self.xstep, j * self.ystep).distance( inter) / step
95                          #if item[0].contains( shp.Point( (i + 0) * self.xstep, (j + 1) * self.ystep)):
96                          if not item[0].disjoint( shp.Point( (i + 0) * self.xstep, (j + 1) * self.ystep)):
97                              # calculate new p value
98                              line = shp.LineString( [( i * self.xstep, j * self.ystep), ( (i + 0) * self.xstep,
99                              inter = item[0].intersection( line)
100                             #p = ( inter.bounds[1] - ( j * self.ystep)) / step
101                             p = shp.Point( i * self.xstep, j * self.ystep).distance( inter) / step
```

```
                        #if item[0].contains( shp.Point( (i + 0) * self.xstep, (j - 1) * self.ystep)):
                        if not item[0].disjoint( shp.Point( (i + 0) * self.xstep, (j - 1) * self.ystep)):
                            # calculate new r value
                            line = shp.LineString( [( i * self.xstep, j * self.ystep), ( (i + 0) * self.xstep,
                            inter = item[0].intersection( line)
                            #r = ( ( j * ystep) - inter.bounds.maxy) / step
                            r = shp.Point( i * self.xstep, j * self.ystep).distance( inter) / step
                    #add point
                    self.st[j][i] = pt( 0.0, p, q, r)

    def calc(self, i, j):
        u =  self.st[j + 0][i - 1].u / \
             ( 1.0 + self.st[j][i].q)
        u += self.st[j - 0][i + 1].u / \
             ( self.st[j][i].q * ( 1.0 + self.st[j][i].q))
        u += self.st[j + 1][i - 0].u / \
             ( self.st[j][i].p * (self.st[j][i].p + self.st[j][i].r))
        u += self.st[j - 1][i + 0].u / \
             ( self.st[j][i].r * (self.st[j][i].p + self.st[j][i].r))
        u *= ( self.st[j][i].p * self.st[j][i].q * self.st[j][i].r)
        u /= ( self.st[j][i].p * self.st[j][i].r + self.st[j][i].q)
        return u

    def liebmann(self, tol=1e-8):
        k = 0
        while True:
            Rm = 0.0
            k += 1
            for j in range( self.J + 1):
                for i in range( self.I + 1):
                    #sprawdzic jaki to punkt
                    if not isinstance( self.st[j][i], metal):
                        v = self.st[j][i].u
                        self.st[j][i].u = self.calc( i, j)
                        R = np.abs(self.st[j][i].u - v)
                        if Rm < R:
                            Rm = R
            if Rm < tol:
                break
        return k

    def liebmann_quater(self, tol=1e-8):
        k = 0
        for j in range( self.J/2 + 1):
            self.st[j][self.I/2+1] = imaginary( self.st[j][self.I/2-1])
        for i in range( self.I/2 + 1):
            self.st[self.J/2+1][i] = imaginary( self.st[self.J/2-1][i])
        while True:
            Rm = 0.0
            k += 1
            for j in range( self.J/2+1):
                for i in range( self.I/2+1):
                    #sprawdzic jaki to punkt
                    if not ( isinstance( self.st[j][i], metal) and \
                             isinstance( self.st[j][i], imaginary)):
                        v = self.st[j][i].u
                        self.st[j][i].u = self.calc( i, j)
                        R = np.abs(self.st[j][i].u - v)
                        if Rm < R:
                            Rm = R
            if Rm < tol:
                break
        for j in range( self.J + 1):
            for i in range( self.I + 1):
                if j < self.J / 2 + 1 and i > self.I / 2:
                    self.st[j][i] = self.st[j][self.I - i]
                elif j > self.J / 2:
                    self.st[j][i] = self.st[self.J - j][i]
        return k

    def impedance( self):
        Eyt = [float for x in xrange( self.I+1)]
        Eyd = [float for x in xrange( self.I+1)]
        Exl = [float for x in xrange( self.J+1)]
        Exr = [float for x in xrange( self.J+1)]

        for j in range( self.J + 1):
            # left boundary
            d1 =      ( self.st[j][0].q * self.xstep)
            d2 = d1 + ( self.st[j][1].q * self.xstep)
            e1st = ( self.st[j][1].u - self.st[j][0].u) / d1
            e2nd = ( self.st[j][2].u - self.st[j][0].u) / d2
            Exl[j] = e1st + ( ( e1st - e2nd) / ( ( d2 / d1) - 1))
            # right boundary
```

```
186             d1 =      ( self.st[j][self.I-1].q * self.xstep)
187             d2 = d1 + ( self.st[j][self.I-2].q * self.xstep)
188             e1st = ( self.st[j][self.I-1].u - self.st[j][self.I-0].u) / d1
189             e2nd = ( self.st[j][self.I-2].u - self.st[j][self.I-0].u) / d2
190             Exr[j] = e1st + ( ( e1st - e2nd) / ( ( d2 / d1) - 1))
191
192         for i in range( self.I + 1):
193             # lower boundry
194             d1 =      ( self.st[0][i].p * self.xstep)
195             d2 = d1 + ( self.st[1][i].p * self.xstep)
196             e1st = ( self.st[1][i].u - self.st[0][i].u) / d1
197             e2nd = ( self.st[2][i].u - self.st[0][i].u) / d2
198             Eyd[i] = e1st + ( ( e1st - e2nd) / ( ( d2 / d1) - 1))
199             # upper boundry
200             d1 =      ( self.st[self.J-0][i].r * self.xstep)
201             d2 = d1 + ( self.st[self.J-1][i].r * self.xstep)
202             e1st = ( self.st[self.J-1][i].u - self.st[self.J-0][i].u) / d1
203             e2nd = ( self.st[self.J-2][i].u - self.st[self.J-0][i].u) / d2
204             Eyt[i] = e1st + ( ( e1st - e2nd) / ( ( d2 / d1) - 1))
205
206         E = integ.simps( Exl, None, self.ystep) + \
207             integ.simps( Exr, None, self.ystep) + \
208             integ.simps( Eyt, None, self.xstep) + \
209             integ.simps( Eyt, None, self.xstep)
210         # print 'Exl = {}; Exr = {}; Eyt = {}; Eyb = {}' \
211         #     .format( integ.simps( Exl, None, self.ystep), \
212         #              integ.simps( Exr, None, self.ystep), \
213         #              integ.simps( Eyt, None, self.xstep), \
214         #              integ.simps( Eyt, None, self.xstep))
215         # print 'E = {}' .format( E)
216         return np.sqrt( ( const.mu_0 * self.mu) / \
217                         ( const.epsilon_0 * self.eps)) * \
218             ( 1 / E)
219
220     def plot( self):
221         z = [[float for x in xrange(self.I+1)] for z in range(self.J+1)]
222         for j in range( self.J + 1):
223             for i in range( self.I + 1):
224                 z[j][i] = self.st[j][i].u
225         cmap = plt.get_cmap( 'jet')
226         plt.imshow( z, origin='lower', interpolation='nearest', cmap=cmap)
227         plt.colorbar()
228         plt.show()
```

## A.3. pum/algorithms.py

```
1  import numpy as np
2
3  def newton_raphson( func, a, b, args=(), tol=1e-8):
4      """
5      Znajduje miejsce zerowe funkcji korzystajac z metody Newtona-Raphsona
6
7      Parameters
8      ----------
9      func : function
10         Funkcja ktorej miejsce zerowe jest poszukiwane
11     a : float
12         Poczatek zakresu na ktorym szukane jest miejsce zerowe
13     b : float
14         Koniec zakresu na ktorym szukane jest miejsce zerowe
15     args : tuple, optional
16         Dodatkowe parametry przekazywane do funkcji
17     tol : float, optional
18         Dopuszczalny blad znalezionego moejsca zerowego
19
20     Returns
21     -------
22     zero : float
23         Szacowana lokacja gdzie wartosc funkcji wynosi 0.
24     """
25     ###
26     # Szukanie punktu startowego
27     ###
28     x0 = a * 1.0 # mnozenie przez 1.0 sprawia, ze zmienna staje sie float
29     delta_x = ( b - a) / 25
30     myargs = (x0,) + args
31     fval_min = np.abs( func( *myargs))
32     zero = x0
33     for x0 in np.arange( a, b, delta_x):
```

```
34          myargs = (x0,) + args
35          fval = np.abs( func( *myargs))
36          if fval < fval_min:
37              fval_min = fval
38              zero = x0
39
40      ###
41      # Wlasciwa metoda newtona
42      ###
43      delta_x = 1e-8
44      error = 1
45      while error > tol:
46          x = zero
47          fd1 = func( *(x - delta_x,) + args)
48          fd2 = func( *(x + delta_x,) + args)
49          fderiv = ( fd2 -fd1) / ( 2 * delta_x)
50          if fderiv == 0:
51              msg = "derivative was zero."
52              warnings.warn(msg, RuntimeWarning)
53          fval = func( *(x,) + args)
54          zero = x - ( fval / fderiv)
55          error = zero - x
56
57      return zero
58
59  def n_fun( q, tol=1e-19):
60      """
61      Zwraca wartosc N potrzebna do wyznaczenia modulu
62
63      Parameters
64      ----------
65      q : float
66          stala modularna
67      tol : float, optional
68          dokladnosc
69
70      Returns
71      ----------
72      n : float
73          Wartosc parametru N.
74      """
75      n = 0.0
76      old = -1.0
77      i = 1
78      while( n - old > tol):
79          old = n
80          n = old + q ** ( i * ( i - 1))
81          i = i + 1
82      return n
83
84  def d_fun( q, tol=1e-19):
85      """
86      Zwraca wartosc D potrzebna do wyznaczenia modulu
87
88      Parameters
89      ----------
90      q : float
91          stala modularna
92      tol : float, optional
93          dokladnosc
94
95      Returns
96      ----------
97      d : float
98          Wartosc parametru D.
99      """
100     d = 0.5
101     old = 0.0
102     i = 1
103     while( d - old > tol):
104         old = d
105         d = old + q ** ( i * i)
106         i = i + 1
107     return d
108
109 def k_int( k, tol=1e-8):
110     old = 0
111     k0 = k
112     k0p = np.sqrt( 1 - ( k ** 2))
113     k_int = ( 1 + k0) / ( 1 + k0p)
114     i = 1
115     while( k_int - old > tol):
116         old = k_int
117         k0  = ( 2 / ( 1 + k0 )) * np.sqrt( k0)
```

```
118             k0p = ( 2 / ( 1 + k0p)) * np.sqrt( k0p)
119             k_int = k_int * ( 1 + k0) / ( 1 + k0p)
120             i += 1
121         return k_int
```

# Bibliografia


[1] S. Rosłoniec, *Wybrane metody numeryczne z przykładami w zadaniach inżynierskich*, 2nd ed. Warszawa: Oficyna Wydawnicza Politechniki Warszawskiej, 2002.

[2] ——, *Liniowe obwody mikrofalowe. Metody Analizy i syntezy*, 1st ed. Warszawa: Wydawnictwo Komunikacji i Łączności, 1999.

[3] ——, *Algorytmy projektowania wybranych liniowych układów mikrofalowych*, 1st ed. Warszawa: Wydawnictwo Komunikacji i Łączności, 1987.