同濟大學

微波与天线大作业

------Smith 圆图软件设计与演示

题目名称	基于Matlab——guide模块的 Smith 圆图软件设计与演示		
学院（系）	电子与信息工程学院		
专　　业	通信工程		
学生姓名	斯提凡	学　号	1656038
学生姓名	谢宸虓	学　号	1652556

2019 年 4 月 12 日

摘要

史密斯圆图(Smith chart)是在反射系散平面上标绘有归一化输入阻抗（或导纳）等值圆族的计算图。是一款用于电机与电子工程学的图表，主要用于传输线的阻抗匹配上。该图由三个圆系构成，用以在传输线和某些波导问题中利用图解法求解。由于一条传输线(transmission line)的电阻抗力(impedance)会随其长度而改变，要设计一套匹配(matching)的线路，需要通过不少繁复的计算程序，所以史密斯圆图成为求解传输线问题的常用工具。

在此篇报告中，我们借助Matlab——guide模块完成Smith圆图软件设计与演示，实现的功能有：通过键盘输入负载阻抗、线长（λ）以及特征阻抗，程序并行输出输入阻抗、输入导纳、反射系数及驻波比等数据结果及动态演示过程。计算模块分为反射系数计算、输入阻抗计算及整个Smith圆图等。Smith画图演示模块分为等归一化电阻圆、等归一化电抗圆、反射系数圆等，实现确定阻抗值在Smith圆图上的位置等功能。之后，我们进行界面排版、计算结果测试及在向实际应用的完善中进行一些调整。

我们的实测结果可以看出 Smith 圆图软件相较于传统纸质圆图的分析过程拥有精确度更高、计算速度更快的优势。Smith圆图相关的一些基本运算在我们的软件中可以得到实现。在画图、算法等方面我们深知存在不足，请老师和同学进行批评和指正，以便于我们完善软件及对背后的原理进行更进一步的理解。

关键词：Smith圆图；guide；Smith圆图软件；传输线理论

目录

第一部分：原理综述

1.1 Smith 圆图原理综述

传输线能引导电磁波向一定方向的传输，为了提高传输效率，需要使传输线负载与特征阻抗匹配。Smith 圆图是一种辅助图形，它在求解传输线问题时是非常有用的。虽然还有一些其他的阻抗和反射系数圆图可以用于类似的问题，但是 Smith 圆图可以说是最知名而且应用最广泛的。【1】Smith 圆图是把特征参数和工作参数形成一体，采用图解法解决的一种专用 chart。它是在 1939 年由 P.Smith 在贝尔电话实验室工作时开发的。Smith 圆图不只是一种图形技术，除了作为微波设计的众多流行的计算机辅助设计（CAD）软件和检测设备中的组成部分外，Smith 圆图还提供了一个使传输线现象可视化的方法，因此，从教学的理念来说，它也是重要的。

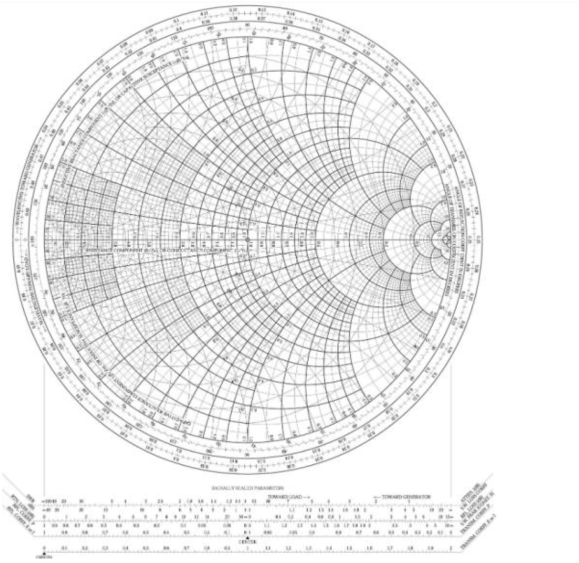


图 1.1 Smith 圆图

Smith 圆图的基本在于算式: $\Gamma = \frac{z_L - 1}{z_L + 1}$，式中的Γ代表其线路的反射系数，zL 是归一化负载阻抗，即 ZL/Z0。其中，ZL 是电路的负载值，Z0 是传输线的特性阻抗值，通常会使用 50Ω。传输线原理是衔接场论分析与基本电路学的桥梁，传输线分为两种状态：行波状态和驻波状态。在行波状态下，传输线终端与其负载电阻完全匹配，传输能量全部被负载吸收，没有能量反射，此时传输线上只有入射波没有反射波，传输效率达到最高。在驻波状态下，传输线终端阻抗不匹配，部分能量反射回来，形成反射波，由于入射波与反射波的相互叠加，而形成了驻波。对于有负载的无耗传输线上负载电流与电压关系，电流与电压为入、反射波之和。

$$V(z)=V_0^+e^{-j\beta z}+V_0^-e^{j\beta z}$$ 式 1.1

$$I(z)=\frac{1}{Z_0}(V_0^+e^{-j\beta z}-V_0^-e^{j\beta z})$$ 式 1.2

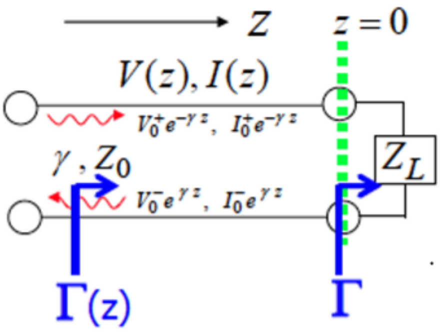

Z
z = 0
V(z), I(z)
$V_0^+e^{-\gamma z}$, $I_0^+e^{-\gamma z}$
γ, Z_0
$V_0^-e^{\gamma z}$, $I_0^-e^{\gamma z}$
Z_L
Γ(z)
Γ

图 1.2 有负载的无耗传输线

由式 1.1 和式 1.2 得，

$$Z_L=\frac{V(z=0)}{I(z=0)}=Z_0\frac{V_0^++V_0^-}{V_0^+-V_0^-}$$ 式 1.3

用入射电压波的振幅归一化反射电压波的振幅，定义为电压反射系数Γ：

$$\Gamma=\frac{V_0^-}{V_0^+}=\frac{Z_L-Z_0}{Z_L+Z_0}$$ 式 1.4

于是线上的总电压和总电流可以写成：

$$V(z)=V_0^+(e^{-j\beta z}+\Gamma e^{j\beta z})$$ 式 1.5

$$I(z)=\frac{V_0^+}{Z_0}(e^{-j\beta z}-\Gamma e^{j\beta z})$$ 式 1.6

从这些表达式可以看出，线上的电压和电流是由入射波和反射波的叠加组成的，这样的波称为驻波。只有当Γ=0 时，才不会有反射波。为了得到Γ=0，阻抗 ZL 必须等于该传输线的特征阻抗 Z0。这样的负载称为传输线的匹配负载。

由式 1.4 得到的反射系数与阻抗的关系：$Z_L=Z_0\frac{1+\Gamma_L}{1-\Gamma_L}$ 就是前文所说的 Smith 圆图的基本算式。

1.2 Smith 圆图基本思想

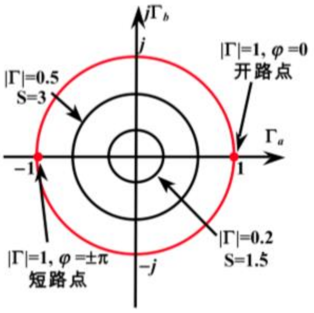
Smith 圆图又称阻抗圆图，其基本思想有三条： 1、特征参数归一思想。这一思想是形成圆图的最关键的一点，包含阻抗归一和电长度归一，即 zL= ZL/Z0 与 $\vartheta=\frac{2\pi}{\lambda_g}l$ 。2、以系统不变量|Γ|作为 Smith 圆图的基底在无耗传输线中，|Γ|是系统的不变量。所以由|Γ|从 0 到 1 的同心圆作为 Smith 圆图的基底，使我们可能在 一有限空间表示全部胡工作参数Γ、Z(Y)等。 3、把阻抗（或导纳），驻波比关系套覆在|Γ|圆上。

使用公式计算往往涉及复数运算，比较麻烦。利用 Smith 圆图可以简便求解，并且容易看出准确结果的趋向，而其作图误差在工程允许的范围内。

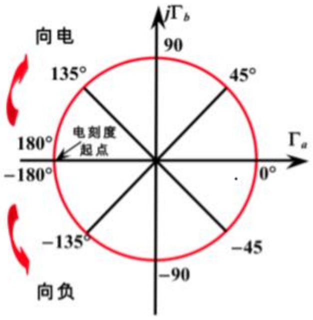
1.3 Smith 圆图构成原理

利用归一化阻抗与反射系数一一对应的关系，将归一化阻抗表示在反射系数复平面上。Smith 圆图的基本构成包括两个曲线坐标系和四组曲线：1、反射系数曲线极坐标（等反射系数模值圆、反射系数相角射线）2、归一化阻抗曲线坐标（等归一化电阻圆、等归一化电抗圆）。

反射系数圆：令Γ(z)= Γa+jΓb 可得Γaˆ2+Γbˆ2=|ΓL|ˆ2 且|ΓL|不大于 1，得到等反射系数模值圆的方程。



$\varphi=\varphi_L-2\beta z'$=tan^(-1)（Γa/Γb)得到反射系数相角射线方程。



特点：绕圆一周相当于考察点在线上移动λ/2。向电源移动，顺时针旋转。向负载移动，逆时针旋转。电刻度起点约定为（-1，0）点。

阻抗圆：rL+jxL=((1+Γr)+jΓi)./((1-Γr)-jΓi)，用其分母的共复轭分别乘以分子和分母，就可以求出上述方程的实部和虚部为：

$$\bar{R}=\frac{1-\Gamma_a^2-\Gamma_b^2}{(1-\Gamma_a^2)^2+\Gamma_b^2}\qquad \bar{X}=\frac{2\Gamma_b}{(1-\Gamma_a^2)^2+\Gamma_b^2}$$

式 1.7 式 1.8

重新整理的等归一化电阻圆方程和等归一化电抗圆方程为：

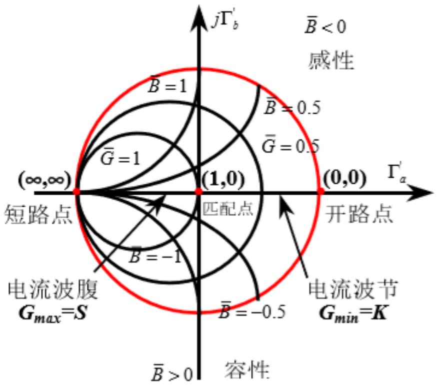
$$\left(\Gamma_a-\frac{\bar{R}}{\bar{R}+1}\right)^2+\Gamma_b^2=\left(\frac{1}{\bar{R}+1}\right)^2$$

$$(\Gamma_a-1)^2+\left(\Gamma_b-\frac{1}{\bar{X}}\right)^2=\left(\frac{1}{\bar{X}}\right)^2$$

式 1.9 式 1.10

电阻圆圆心坐标：$(\frac{\bar{R}}{1+\bar{R}},0)$，半径$\frac{1}{1+\bar{R}}$，所有的电阻圆都在（1，0）点相切。阻抗圆圆心坐标：$(1,\frac{1}{\bar{X}})$，半径$\frac{1}{|\bar{X}|}$，所有的阻抗圆都在（1，0）点与实轴相切。

导纳圆图：$\Gamma=(z-1)/(z+1)=(1-y)/(1+y)$，公式在形式上相同，所以导纳圆图与阻抗原图在形式上及数值上有紧密联系，同一张阻抗圆图可以被用来当作导纳圆图使用。



1.4 GUI

图形用户界面（Graphical User Interface，简称 GUI，又称图形用户接口）是指采用图形方式显示的计算机操作用户界面。与早期计算机使用的命令行界面相比，图形界面对于用户来说在视觉上更易于接受。然而这界面若要通过在显示屏的特定位置，以”各种美观而不单调的视觉消息“提示用户”状态的改变，势必得比简单的消息呈现花上更多的计算能力。

图形用户界面是一种人与计算机通信的界面显示格式，允许用户使用鼠标等输入设备操纵屏幕上的图标或菜单选项，以选择命令、调用文件、启动程序或执行其它一些日常任务。与通过键盘输入文本或字符命令来完成例行任务的字符界面相比，图形用户界面有许多优点。图形用户界面由窗口、下拉菜单、对话框及其相应的控制机制构成，在各种新式应用程序中都是标准化的，即相同的操作总是以同样的方式来完成，在图形用户界面，用户看到和操作的都是图形对象，应用的是计算机图形学的技术。【2】

Matlab 的命令行中输入 guide 调用 gui 模块，模块提供众多模块属性选项如：按钮、文本框、滑轨等，我们通过改变每一个模块的 callback 函数即可实现我们想要达到的功能。回调函数（callback）中使用 set、get 等简单的函数实现我们需要在回调函数中进行的操作。对于 guide 自动生成的头部代码，我们不需要进行改动，简化了我们制作窗口界面的过程。

第二部分：问题简要描述及分析

2.1 问题简要描述

SMITH 圆图的软件设计，计算结果以图形和数据并行输出，动态显示运用圆图进行计算的全部过程。整个软件分为用户图形界面、SMITH 圆图计算和 SMITH 圆图演示三个模块。SMITH 圆图计算模块又分为反射系数计算、输入阻抗计算及整个 SMITH 圆图； SMITH 画图演示模块分为等归一化电阻圆、等归一化电抗圆、反射系数圆等；确定阻抗值在 SMITH 圆图上的位置等功能。

2.2 分析

2.2.1 界面要素

根据我们要实现的功能，界面至少应包括：运行按钮、图像、输入输出文本框。

2.2.2 计算方法

计算模块要实现的功能可以简述为：通过输入的负载阻抗、特征阻抗、线长，输出输入阻抗、输入导纳、反射系数、驻波比。计算方法背后的理论我们在报告的前面有提及，接下来列出具体的数学算式。

归一化阻抗 zin=负载阻抗 Zin/特征阻抗 Z0；

驻波情况下，任一点的输入阻抗会发生改变：

$$Z_{in}(-\ell)=\frac{V(-\ell)}{I(-\ell)}=Z_o\frac{V_o^+e^{j\beta\ell}+V_o^-e^{-j\beta\ell}}{V_o^+e^{j\beta\ell}-V_o^-e^{-j\beta\ell}}=Z_o\frac{1+\Gamma e^{-j2\beta\ell}}{1-\Gamma e^{-j2\beta\ell}}$$

式 2.1；

输入导纳是输入阻抗的倒数；

驻波比（SWR）=（1+|Γ|）/（1-|Γ|）;

2.2.3 Smith 圆图绘制

Smith 圆图绘制分为：坐标系建立、画最大外圆、标注虚轴刻度、画电阻圆族、画电抗圆族以及数值标注。画图部分计划使用 plot 函数实现，标注文本部分计划使用 text 函数实现，应注意的点是因为图形是圆形，文本标注时应增加角度设定。

Smith 圆图线条虽然复杂，但仔细研究会发现其内在的规律。圆图中线条并非均匀分布，我们需要设定一定的界限决定不同的区域的线条疏密情况。

2.2.4 Smith 圆图软件动画演示

Smith 圆图动画演示分为：λ=0 的情况下计算出归一化电阻时“点”的标注（可采用*、o、或者 text 函数标注坐标的方法）、画反射系数圆、画原点到第

一个标注点的射线、旋转射线后得到的直线及标注两个交点。为了显示出标注的顺序，我们想到了两种方式：1、每一项计算单独设置运行按钮，优点是我们可以较明确的观察每一项计算涉及的动画，缺点是按键次数较多，每计算一项就要按一次按钮，这对于计算延时并不高的程序来说是存在冲突的。因为我们认为人为操作所导致的延时是会大于我们的程序计算延时的。2、只设置一个运行按钮，动画通过设置 pause 函数依次输出。优点是符合要求且操作方便，缺点是相较于前一种方式不易于观察每一项动画。

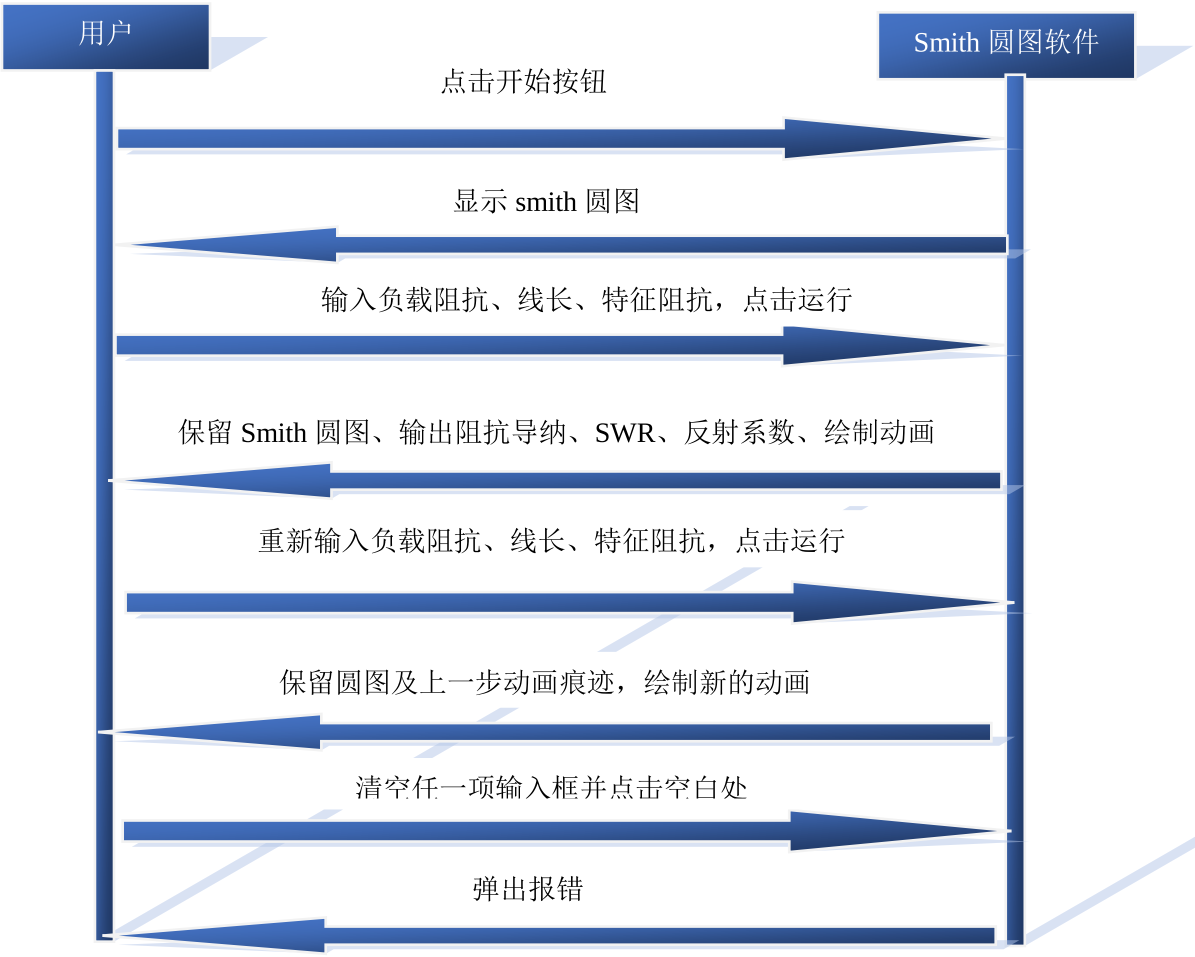
2.2.5 Smith 圆图软件排版

出于美观简洁的角度，我们不设置任何和计算以及演示无关的功能（例如时间等），并且我们默认使用者熟悉 Smith 圆图的基本原理而不再分页制作说明框（我们的按钮以及静态文本框的 string 属性可以让使用者明白其功能是什么）。

排版将图像显示与文本类对象分隔开，采用左右结构（图在左，其余在有），我们更倾向于设计横板的软件界面。

第三部分：设计流程

3.1 关于“用户、Smith 圆图软件”两者之间的事件跟踪图及流程图


用户
Smith 圆图软件
点击开始按钮
显示 smith 圆图
输入负载阻抗、线长、特征阻抗，点击运行
保留 Smith 圆图、输出阻抗导纳、SWR、反射系数、绘制动画
重新输入负载阻抗、线长、特征阻抗，点击运行
保留圆图及上一步动画痕迹，绘制新的动画
清空任一项输入框并点击空白处
弹出报错

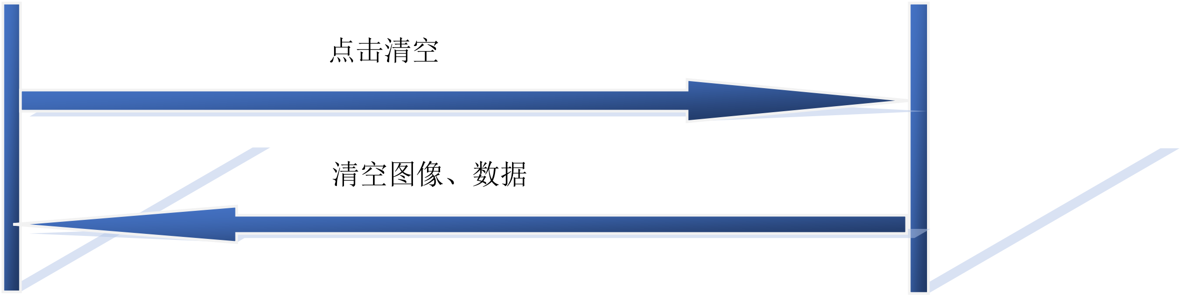


图 3.1 事件跟踪图

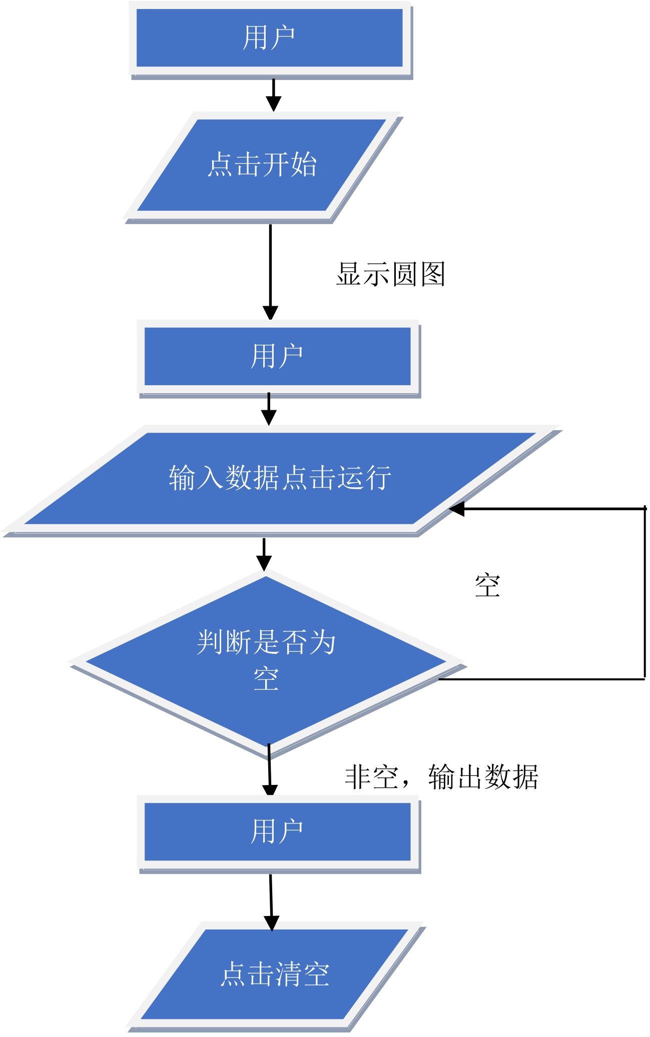


图 3.2 流程图

3.2 软件界面详细组成

软件界面要素有：Smith 圆图（静态文本）、axe 图像输出部分、负载实部虚部、线长、特征阻抗、输入阻抗、输入导纳、SWR、反射系数、反射系数（模）、开始运行清空按钮以及单位。界面如图 3.2 所示。

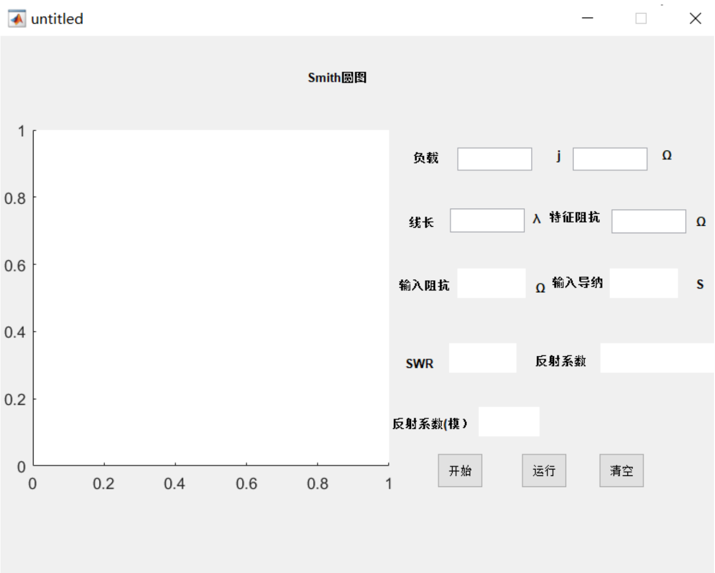


图 3.2 界面

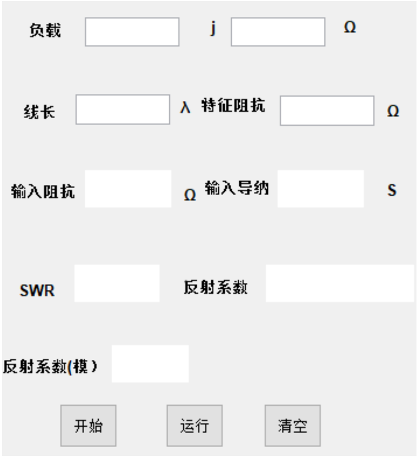


图 3.3

3.3 pushbutton 回调函数的设计

对应软件部分为图 3.3 所示。

3.3.1 开始按钮

回顾问题描述及分析，软件将在开始按钮按下后绘制出 Smith 圆图，所以开始按钮的回调函数的主要组成部分是绘制 Smith 圆图的代码。Smith 圆图绘制分为：坐标系建立、画最大外圆、标注虚轴刻度、画电阻圆族、画电抗圆族以及数值标注。接下来一一说明实现的过程。

（1）画最大外圆及标注虚轴刻度的代码如下：

```
axis square;
wavelengths=0:.01:.5;
angle = linspace(pi,-pi,length(wavelengths));
wave_circle = 1.05*exp(1i*angle);
plot(real(wave_circle),imag(wave_circle),’r’);
```

```
hold on
plot([-1.05 1.05],[0 0],'color','r');
for i=1:length(wavelengths)-1
    x=real(1.025*exp(1i*angle(i)));
    y=imag(1.025*exp(1i*angle(i)));
    if(x>0)
         rot_angle=atan(y/x)*180/pi-90;
    else
         rot_angle=atan(y/x)*180/pi+90;
    end
    if(wavelengths(i)==0)
         word = '0.00';
    elseif(mod(wavelengths(i),.1)==0)
         word =[num2str(wavelengths(i)) '0'];
    else
         word = num2str(wavelengths(i));
    end
    text(x,y,word,'Rotation',rot_angle)
end
hold on
```

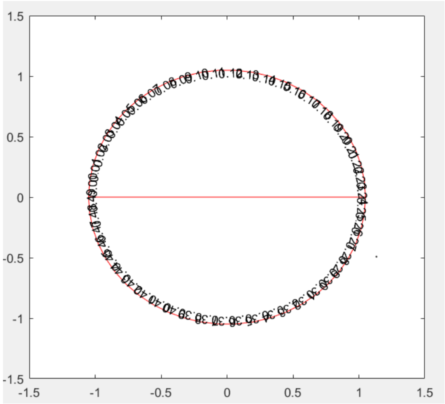


图 3.4 画最大外圆及标注虚轴刻度

（2）画阻抗圆实数部分（代码见附件）。程序中画图代码的简化版本如下：

```
interval=[(.01:.01:.2),(.22:.02:.5),(.55:.05:1),(1.1:.1:2),(2
.2:.2: 5),(6:10),(12:2:20),(30:10:50)];
alpha=2*pi*(0:0.005:1);
for rr=interval
xc=rr/(1+rr);
rd=1-xc;
plot(xc+rd*cos(alpha),rd*sin(alpha),color);
end
```

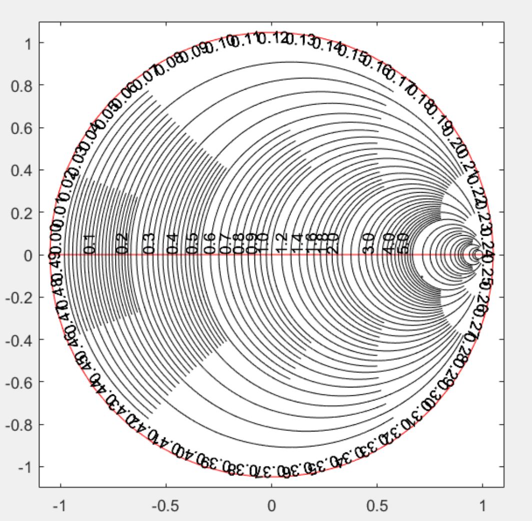


图 3.5 画阻抗圆实数部分

（3）画阻抗圆虚数部分（代码见附件）。程序中画图代码的简化版本如下：

```
interval2=[(.01:.01:.5),(.55:.05:1),(1.1:.1:2),(2.2:.2:5),(6:10),(12:2:20),(30:10:50)];
 for xx=interval2
 xc=1;     % 圆心 x 的坐标
 yc=1/xx; % 圆心 y 的坐标
 rd=1/xx; % 圆的半径
 alpha_xx= 2*atan(xx)*(0:0.005:1);
plot(xc-rd*sin(alpha_xx),yc-rd*cos(alpha_xx),’color’,chart_color);
plot(xc-rd*sin(alpha_xx),-yc+rd*cos(alpha_xx),’color’,chart_color);
 end
```

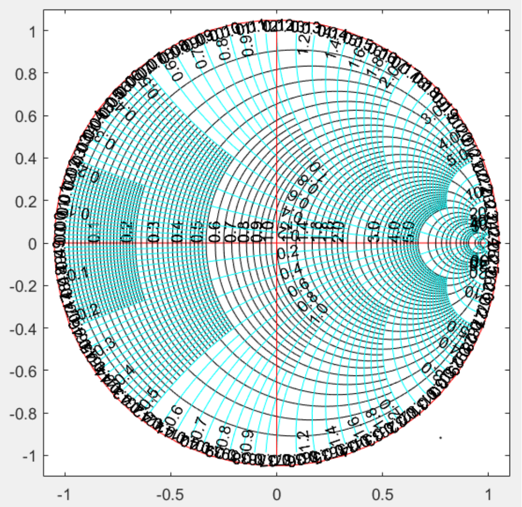


图 3.6 画阻抗圆虚数部分

3.3.2 运行按钮

回顾问题描述及分析，软件在输入负载阻抗、线长、特征阻抗后点击运行按钮，软件将输出输入阻抗、输入导纳、SWR、反射系数，并且进行动画演示。接下来对各自的算法进行说明。

（1）计算反射系数通过代码：

```
z1 = str2double(get(handles.edit9,'string'));
z2 = str2double(get(handles.edit10,'string'));
z3 = str2double(get(handles.edit2,'string'));
d = str2double(get(handles.edit1,'string'));
z = ((z1+j*z2)./z3+j*tan(2*pi*d))/(1+j*((z1+j*z2)./z3)*tan(2*pi*d));
z22=z*z3;
y = 1/z22;
aida = (z-1)/(z+1);
```

aida 的值即为反射系数的值，依据式 2.1。代码中 d 为线长，z1、z2 分别为负载实部与虚部，z3 为特征阻抗。

（2）计算 SWR 通过代码：

```
r = str2double(get(handles.text24,'string'));
r1 = real(r);
r2 = imag(r);
r3 = sqrt((r1)^2+(r2)^2);
r4 = (1+r3)./(1-r3);
```

通过反射系数求得。

（3）计算输入阻抗通过代码：

```
z = ((z1+j*z2)./z3+j*tan(2*pi*d))/(1+j*((z1+j*z2)./z3)*tan(2*pi*d));
```

（4）计算输入导纳通过求输入阻抗的倒数。

（5)动画演示过程通过计算过程中求得的值与其对应的坐标进行找点连线，分为：λ=0 的情况下计算出归一化电阻时“点”的标注（采用 text 函数标注坐标的方法)、画反射系数圆、画原点到第一个标注点的射线、旋转射线后得到的直线及标注两个交点。每一步动画后设置 0.5s 延时。实现了结果动画并行输入，并且可以看到每一步动画的单独演示。

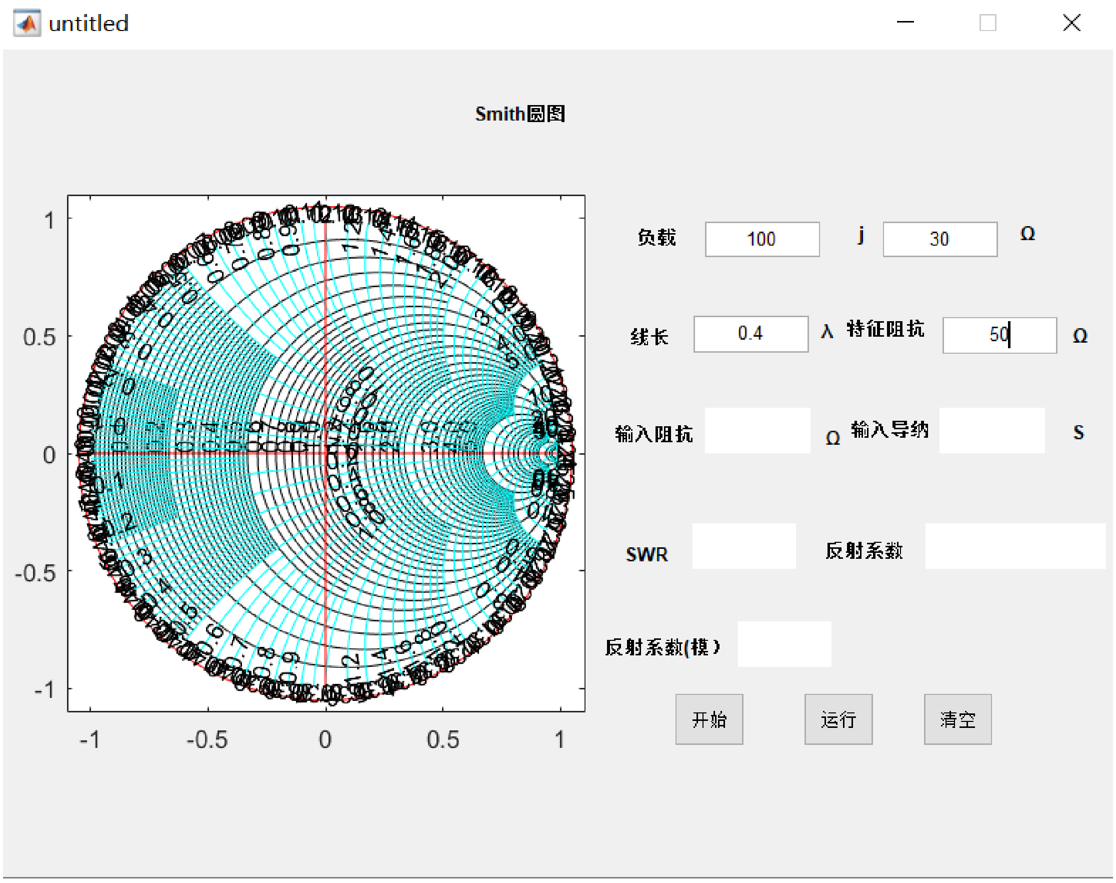
3.3.3 清空按钮

清空按钮实现的功能是点击后清空文本框内容、图像及所有动画演示效果。代码为：

```
set(handles.text21,'string','')
set(handles.text19,'string','')
set(handles.text23,'string','')
set(handles.text24,’string’,’’)
set(handles.text26,’string’,’’)
set(handles.edit9,’string’,’’)
set(handles.edit10,’string’,’’)
set(handles.edit1,’string’,’’)
set(handles.edit2,’string’,’’)
axes(handles.axes1);
cla reset %清空图像
```

第四部分：仿真图示

4.1 第一组



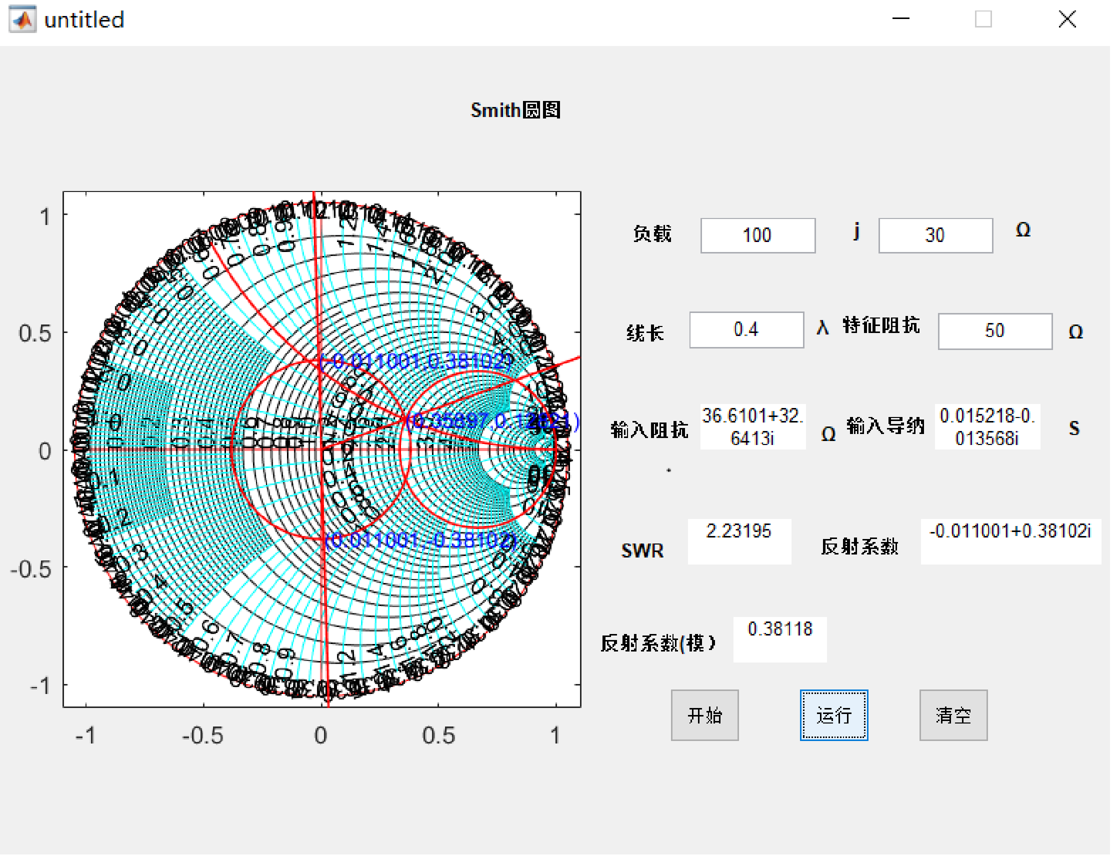
负载 100 j 30 Ω

线长 0.4 λ 特征阻抗 50 Ω

输入阻抗 Ω 输入导纳 S

SWR 反射系数

反射系数(模)



负载 100 j 30 Ω

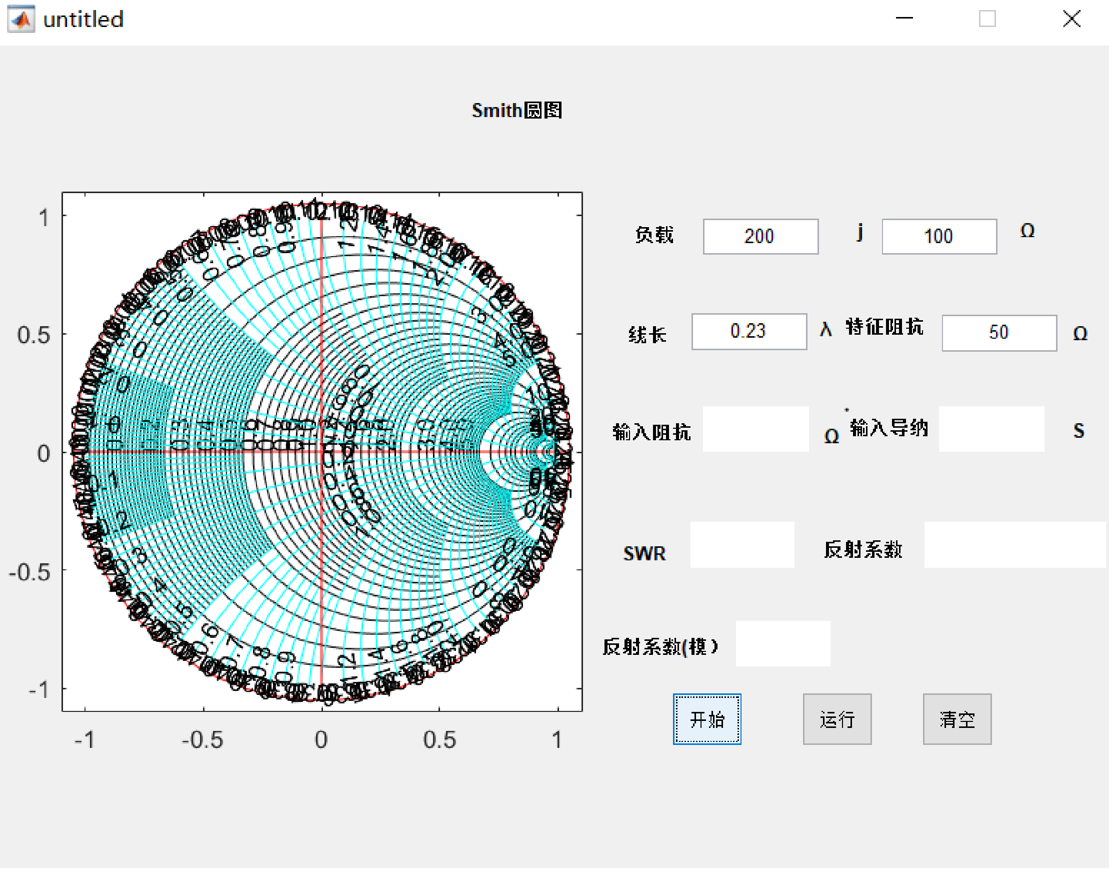
线长 0.4 λ 特征阻抗 50 Ω

输入阻抗 36.6101+32.6413i Ω 输入导纳 0.015218-0.013568i S

SWR 2.23195 反射系数 -0.011001+0.38102i

反射系数(模） 0.38118

4.2 第二组



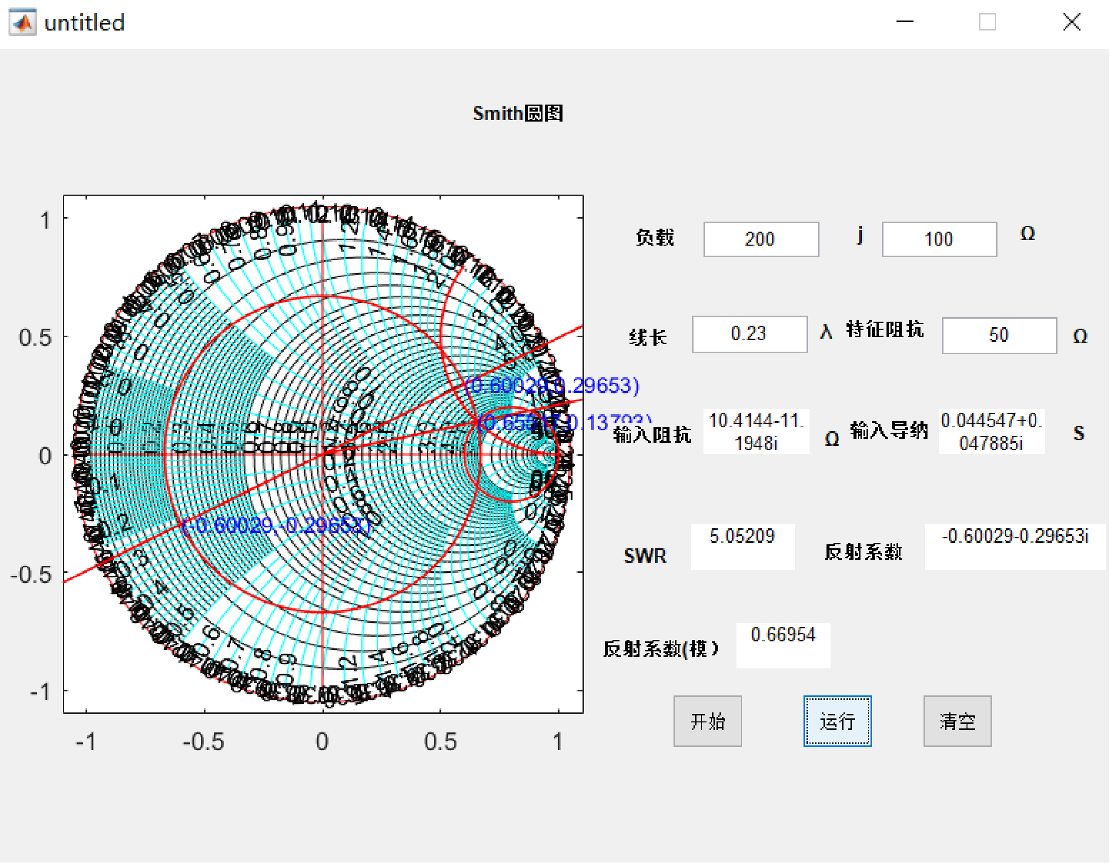
负载 200 j 100 Ω

线长 0.23 λ 特征阻抗 50 Ω

输入阻抗 Ω 输入导纳 S

SWR 反射系数

反射系数(模）



untitled

Smith圆图

负载 200 j 100 Ω

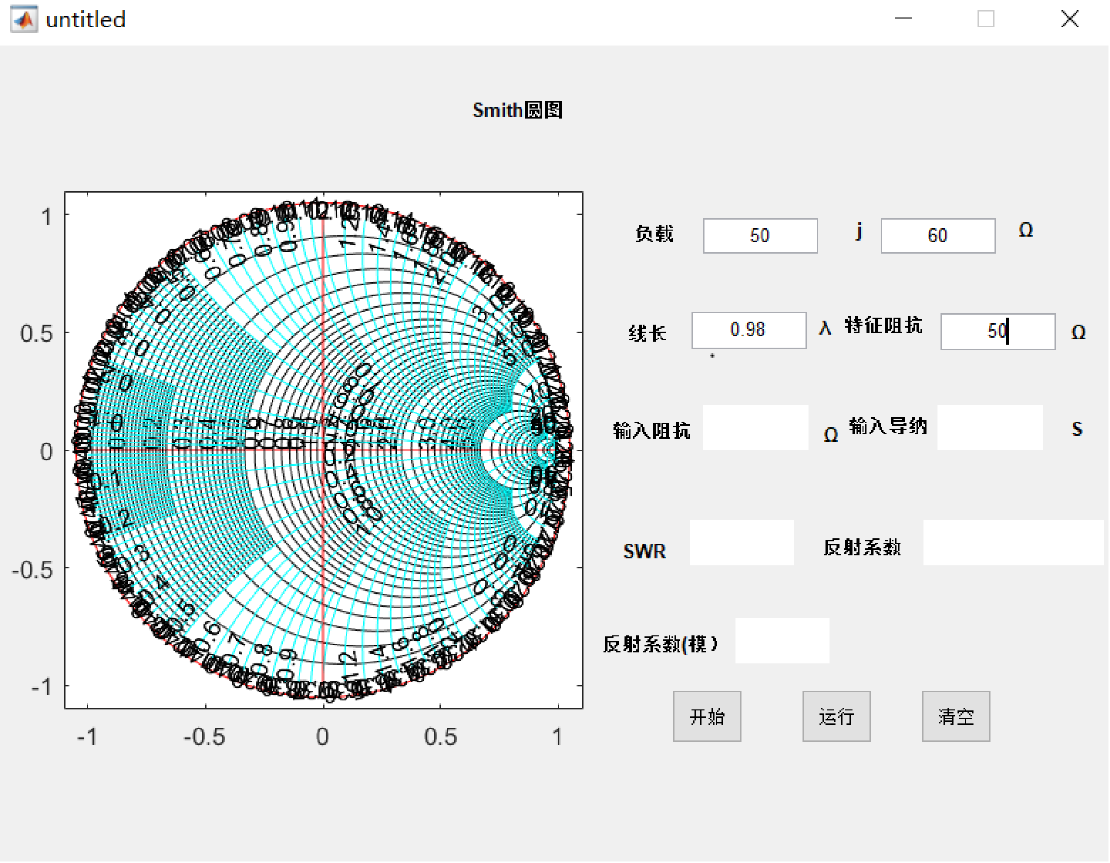
线长 0.23 λ 特征阻抗 50 Ω

输入阻抗 10.4144-11.1948i Ω 输入导纳 0.044547+0.047885i S

SWR 5.05209 反射系数 -0.60029-0.29653i

反射系数(模) 0.66954

4.3 第三组



untitled

Smith圆图

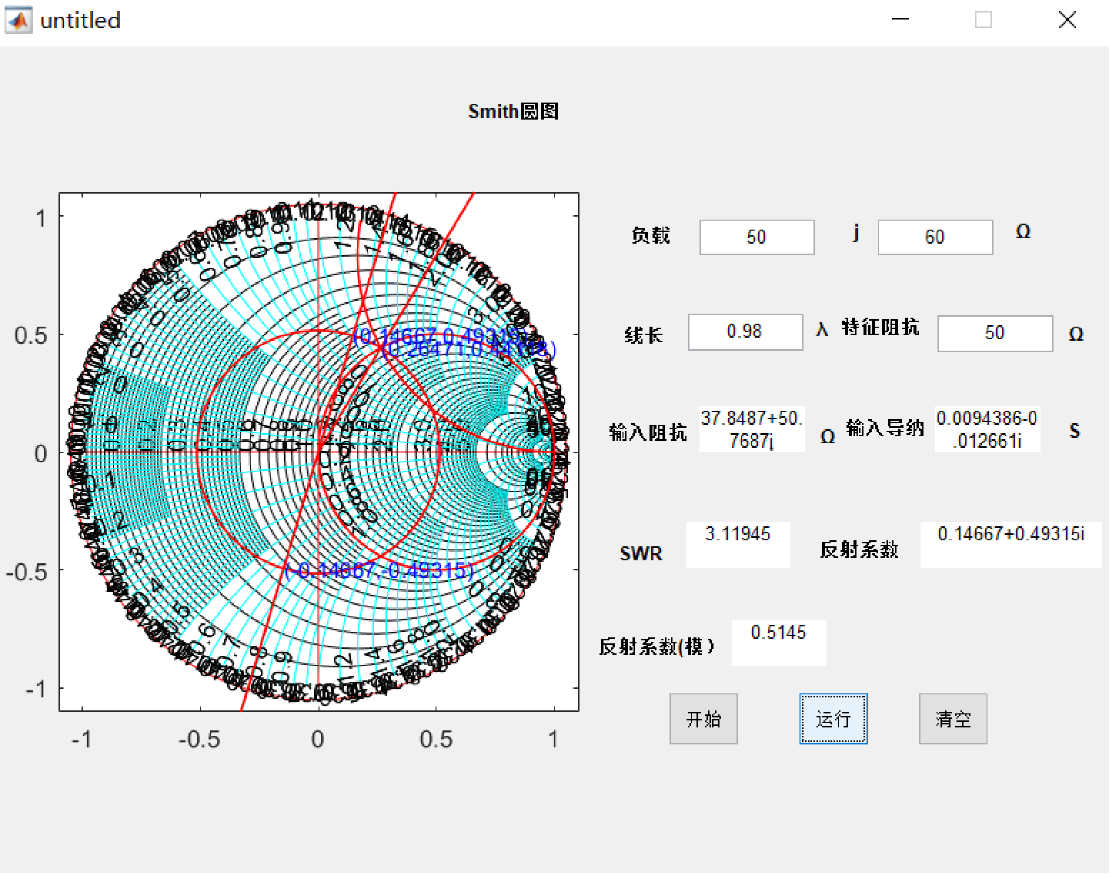
负载 50 j 60 Ω

线长 0.98 λ 特征阻抗 50 Ω

输入阻抗 Ω 输入导纳 S

SWR 反射系数

反射系数(模)



负载 50 j 60 Ω

线长 0.98 λ 特征阻抗 50 Ω

输入阻抗 37.8487+50.7687i Ω 输入导纳 0.0094386-0.012661i S

SWR 3.11945 反射系数 0.14667+0.49315i

反射系数(模) 0.5145

第五部分：源代码

```
function varargout = smith(varargin)
% UNTITLED MATLAB code for untitled.fig
%        UNTITLED, by itself, creates a new UNTITLED or raises the existing
%        singleton*.
%
%        H = UNTITLED returns the handle to a new UNTITLED or the handle to
%        the existing singleton*.
%
%        UNTITLED('CALLBACK',hObject,eventData,handles,...) calls the local
%        function named CALLBACK in UNTITLED.M with the given input arguments.
%
%        UNTITLED('Property','Value',...) creates a new UNTITLED or raises the
%        existing singleton*.   Starting from the left, property value pairs are
%        applied to the GUI before untitled_OpeningFcn gets called.   An
%        unrecognized property name or invalid value makes property application
%        stop.   All inputs are passed to untitled_OpeningFcn via varargin.
%
%        *See GUI Options on GUIDE's Tools menu.   Choose "GUI allows only one
%        instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help untitled
% Last Modified by GUIDE v2.5 20-Apr-2019 18:11:21
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',          mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @untitled_OpeningFcn, ...
                   'gui_OutputFcn',  @untitled_OutputFcn, ...
```

```
                              'gui_LayoutFcn',    [] , ...
                              'gui_Callback',     []);
if nargin && ischar(varargin{1})
        gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
        [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
        gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before untitled is made visible.
function untitled_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject      handle to figure
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
% varargin     command line arguments to untitled (see VARARGIN)

% Choose default command line output for untitled
handles.output = hObject;

%equations were derived using the symbolic toolbox as follows
%solve('R=(1-Gr^2-Gi^2)/((1-Gr)^2+Gi^2)','Gi')
%bound was derived as follows
%solve('1/(R+1)*(-(R+1)*(R-2*R*Gr+R.*Gr^2-1+Gr^2))^(1/2)=0','Gr')
% Update handles structure
guidata(hObject, handles);
global phaseAngle MAX Gr

% UIWAIT makes untitled wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% --- Outputs from this function are returned to the command line.
function varargout = untitled_OutputFcn(hObject, eventdata, handles)
% varargout    cell array for returning output args (see VARARGOUT);
% hObject      handle to figure
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes during object creation, after setting all properties.
function axes1_CreateFcn(hObject, eventdata, handles)
% hObject      handle to axes1 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns called
% Hint: place code in OpeningFcn to populate axes1
% --- Executes on button press in pushbutton1.

function pushbutton2_Callback(hObject, eventdata, handles)
% hObject      handle to pushbutton2 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
%画最大外圆、标注虚轴刻度
        axis square;
        wavelengths=0:.01:.5;
        angle = linspace(pi,-pi,length(wavelengths));
        wave_circle = 1.05*exp(1i*angle);
        plot(real(wave_circle),imag(wave_circle),'r');
```

```
    hold on
    plot([-1.05 1.05],[0 0],'color','r');
    for i=1:length(wavelengths)-1
        x=real(1.025*exp(1i*angle(i)));
        y=imag(1.025*exp(1i*angle(i)));
        if(x>0)
            rot_angle=atan(y/x)*180/pi-90;
        else
            rot_angle=atan(y/x)*180/pi+90;
        end
        if(wavelengths(i)==0)
            word = '0.00';
        elseif(mod(wavelengths(i),.1)==0)
            word =[num2str(wavelengths(i)) '0'];
        else
            word = num2str(wavelengths(i));
        end
        text(x,y,word,'Rotation',rot_angle)
    end
    hold on
    bound2=0;
    bound3=0;
    min_bound1=0;
    min_bound2=0;
    max_bound2=0;
    interval=[.01:.01:.2,.22:.02:.5,.55:.05:1,1.1:.1:2 , 2.2:.2:5 , 6:10 , 12:2:20 , 30:10:50];
    interval2=[.01:.01:.5 , .55:.05:1 , 1.1:.1:2 , 2.2:.2:5 , 6:10 , 12:2:20 , 30:10:50];
% 画电阻圆族
U = @(R,X)(R^2+X^2-1)/(R^2+2*R+1+X^2);
V = @(R,X)(2*X)/(R^2+2*R+1+X^2);
MAX = 2019
Gr = linspace(-1,1,MAX)
axis square
axis([-1.1,1.1,-1.1,1.1]);
hold on
for R = interval
    min_bound1 = (R-1)/(R+1);
    if (R<0.2)
        if (mod(R,0.1) == 0)
            max_bound = U(R,2)
        elseif (mod(R,0.02) == 0)
            max_bound = U(R,0.5)
        else
            max_bound = U(R,0.2)
        end
        if(R == 0.05 || (R<0.151 && R>0.149))
            min_bound2 = U(R,0.5)
            max_bound2 = U(R,1)
        end
    elseif (R<0.5)
        if (mod(R,0.2) == 0)
            max_bound = U(R,5)
        elseif (mod(R,0.1) == 0)
            max_bound = U(R,2)
        elseif (R == 0.55 || R == 0.65 || R == 0.75)
            temp = U(R,0.5)
            min_bound2 = max(min_bound1,temp)
            max_bound = U(R,1)
        else
            max_bound = U(R,0.5)
        end
    elseif (R<1)
        if (mod(R,0.2) == 0)
            max_bound = U(R,5)
        elseif (mod(R,0.1) == 0)
            max_bound = U(R,2)
        elseif (R == 0.25 || R == 0.35 || R == 0.45)
            temp = U(R,0.5)
            min_bound2 = max(min_bound1,temp)
            max_bound = U(R,1)
        elseif (R<0.5)
            max_bound = U(R,0.5)
        else
            max_bound = U(R,1)
        end
    elseif (R<5)
        if (mode(R,2) == 0)
            max_bound = U(R,20)
        elseif (mod(R,1) == 0)
            max_bound = U(R,10)
        elseif (R>2)
            max_bound = U(R,5)
        else
            if (mod(R,0.2) == 0)
                max_bound = U(R,5)
            else
                max_bound = U(R,2)
            end
        end
    elseif (R<10)
        if (mod(R,2) == 0)
            max_bound = U(R,20)
        else
            max_bound = U(R,10)
        end
    else
        if (R == 10 || R == 20)
            max_bound = U(R,50)
        elseif (R == 50)
            max_bound = 1
        elseif (R<20)
            max_bound = U(R,20)
        else
            max_bound = U(R,50)
        end
    end
    index = ceil((min_bound1 + 1)*(MAX - 1)/2 + 1);
    actual_value = Gr(index);
    if (actual_value<min_bound1)
        index = index + 1;
    end
    MIN=index;
    index = ceil((max_bound + 1)*(MAX - 1)/2 + 1);
    actual_value = Gr(index);
    if (actual_value>max_bound)
        index = index - 1;
    end
    MIN2 = ceil((min_bound2 + 1)*(MAX - 1)/2 + 1);
    actual_value = Gr(MIN2);
    if (actual_value<min_bound2)
```

```
        MIN2 = MIN2 + 1;
    end
    MAX2 = ceil((max_bound2+1)*(MAX-1)/ 2+1);
    actual_value = Gr(MAX2);
    if(actual_value<max_bound2)
        MAX2 = MAX2 + 1;
    end
    r_L_a=1/(R+1)*(-(R+1)*(R-2*R.*Gr(MIN:index)+
R.*Gr(MIN:index).^2-1+Gr(MIN:index).^2)).^(1/ 2 );
    r_L_b=-1/(R+1)*(-(R+1)*(R-2*R.*Gr(MIN:index)+
R.*Gr(MIN:index).^2-1+Gr(MIN:index).^2)).^(1/ 2 );
    r_L_b(1)=0;
    r_L_a(1)=0;

r_L_a2=1/(R+1)*(-(R+1)*(R-2*R.*Gr(MIN2:MAX2 )+R.*Gr(MIN2:MAX2).^2-1+Gr(MIN2:MA
X2).^2)).^(1/2);

r_L_b2=-1/(R+1)*(-(R+1)*(R-2*R.*Gr(MIN2:MAX2)+R.*Gr(MIN2:MAX2).^2-1+Gr(MIN2:M
AX2).^2 )).^(1/2);
    r_L_a3=1/(R+1)*(-(R+1)*(R-2*R.*Gr(MIN2:index)
+R.*Gr(MIN2:index).^2-1+Gr(MIN2:index).^2)).^(1/2);

r_L_b3=-1/(R+1)*(-(R+1)*(R-2*R.*Gr(MIN2:index)+R.*Gr(MIN2:index).^2-1+Gr(MIN2:index)
.^2)).^(1/2);
    if(~(R>.2 && R<.5 && ~(mod(R,.02)==0)))
            color ='k';
    end
    plot(Gr(MIN:index),r_L_a(1:index-MIN+1),color,Gr(MIN:index),
r_L_b(1:index-MIN+1),color);
    if(R<=1)
        if(mod(R,1)==0)
            word = [num2str(R) '.0'];
        else
            word = num2str(R);
        end
        if(mod(R,.1)==0)

set(text(Gr(MIN),0,word),'Rotation',90,'HorizontalAlignment','left','VerticalAlignment','bott om');
        end
    elseif(R<=2)
        if(mod(R,.2)==0)
            if(mod(R,1)==0)
                word = [num2str(R) '.0'];
            else
                word = num2str(R);
            end

set(text(Gr(MIN),0,word),'Rotation',90,'HorizontalAlignment','left','VerticalAlignment','bott om');
        end
    elseif(R<=5)
        if(mod(R,1)==0)
            set(text(Gr(MIN),0,[num2str(R)
'.0']),'Rotation',90,'HorizontalAlignment','left','VerticalAlignment','bottom');
        elseif(mod(R,10)==0)

set(text(Gr(MIN),0,num2str(R)),'Rotation',90,'HorizontalAlignment','left','VerticalAlignment' ,'bott
om');
        end
    elseif(R == 0.25 || R == 0.35 || R==0.45)
        plot(Gr(MIN2:index),r_L_a3,'b');
        plot(Gr(MIN2:index),r_L_b3,'b');
    elseif(R==.05 || (R>.149 && R<.151))

plot(Gr(MIN2:MAX2),r_L_a2(length(Gr(MIN2:MAX2))-length(r_L_a2)+1:length(r_L_a2)),'b');

plot(Gr(MIN2:MAX2),r_L_b2(length(Gr(MIN2:MAX2))-length(r_L_b2)+1:length(r_L_b2)),'b');
    end
end
%%%%%%
    hold on
    bound2=0;
    bound3=0;
    inter_bound=0;
    imag_bound=0;
    imag_bound_y=0;
    imag_rad=0;
    max_bound=0;
    inter_index=0;
    imag_index=0;
    phaseAngle=0;
    for X = interval
    inter_bound = (-1+X^2)/(X^2+1);
    imag_bound = (-1+X)/X;
    angle_point = 0;
    if(inter_bound ~= 0)
        angle_point = sqrt(1-inter_bound^2)/inter_bound;
    end
    imag_bound_y = 1/2/X*(-2+2*(1-X^2+2*X^2.*inter_bound-X^2.*inter_bound.^2).^(1/2));
    imag_rad = (imag_bound^2 + imag_bound_y^2)^(1/2);
    condition = imag_rad < 1;
    if(inter_bound > 1)
        inter_bound = 1;
    elseif(inter_bound < -1)
        imag_bound=-1;
    end
    if(imag_bound > 1)
        imag_bound = 1;
    elseif(imag_bound < -1)
        imag_bound=-1;
    end
    if(X<0.2)
        if(mod(X,.1)==0)
            max_bound = U(2,X)
        elseif(mod(X,.02)==0)
            max_bound = U(0.5,X)
        else
            max_bound = U(0.2,X)
        end
    elseif(X<1)
        if(mod(X,0.2) == 0)
            max_bound = U(5,X)
        elseif(mod(X,0.1) == 0)
            max_bound = U(2,X)
        elseif(x<0.5)
            max_bound = U(0.5,X)
        else
            max_bound = U(1,X)
        end
    elseif(X<5)
        if(mod(X,2) == 0)
```

```
            max_bound = U(20,X)
        elseif(mod(X,1) == 0)
            max_bound = U(10,X)
        elseif(x>2)
            max_bound = U(5,X)
        else
            if(mod(X,0.2) == 0)
                max_bound = U(5,X)
            else
                max_bound = U(2,X)
            end
        end
    elseif(X<10)
        if(mod(X,2) == 0)
            max_bound = U(20,X)
        else
            max_bound = U(10,X)
        end
    else
        if(X == 10 || X == 20)
            max_bound = U(50,X)
        elseif(X == 50)
            max_bound = 1
        elseif(X<20)
            max_bound = U(20,X)
        else
            max_bound = U(50,X)
        end
    end
    inter_index = ceil((inter_bound+1)*(MAX-1)/ 2+1);
    imag_index = ceil((imag_bound+1)*(MAX-1)/ 2+1);
    index4 = ceil((max_bound+1)*(MAX-1)/ 2+1);
    index1 = max(inter_index,imag_index);
    index2 = min(imag_index,inter_index);
    if(condition)
        index3=imag_index;
    else
        index3=inter_index;
    end
    actual_value1 = Gr(index1);
    actual_value2 = Gr(index2);
    actual_value3 = Gr(index3);
    actual_value4 = Gr(index4);
    if((actual_value1 > inter_bound && index1 == inter_index)||(actual_value1 > imag_bound
&& index1 == imag_index))
        index1 = index1 - 1;
    end
    if((actual_value2 < inter_bound && index2 == inter_index)||(actual_value2 < imag_bound
&& index2 == imag_index))
        index2 = index2 + 1;
    end
    if((actual_value3 < inter_bound && index3 == inter_index)||(actual_value3 < imag_bound
&& index3 == imag_index))
        index3 = index3 + 1;
    end
    if(actual_value4 > max_bound)
        index4 = index4 - 1;
    end
    MIN = index2;
    MAX2 = index1;
    MAX3 = index4;
    MIN2 = index3;
    x_L_a =
real(1/2/X*(-2+2*(1-X^2+2*X^2.*Gr(MIN2:MAX3 )-X^2.*Gr(MIN2:MAX3).^2).^(1/ 2)));
    x_L_b =
real(1/2/X*(2-2*(1-X^2+2*X^2.*Gr(MIN2:MAX3)-X^2.*Gr(MIN2:MAX3).^2).^(1 /2)));
    x_L_c= real(1/2/X*(2+2*(1-X^2+2*X^2.*Gr(MIN:MAX2)-X^2.*Gr(MIN:MAX2).^2).^(1/
2)));
    x_L_d= real(1/2/X*(-2-2*(1-X^2+2*X^2.*Gr(MIN:MAX2)-X^2.*Gr(MIN:MAX2).^2).^(1/
2)));
    if(MIN2<MAX3)
        x_L_c(1)=x_L_b(1);
        x_L_d(1)=x_L_a(1);
    end
    check1 = abs(round(10000*1
/2/X*(-2-2*(1-X^2+2*X^2*inter_bound-X^2*inter_bound^2)^(1/2))));
    check2 = abs(round(10000*(1-inter_bound^2)^(1 /2)));
    if(imag_bound > -1 && check1 == check2)
        plot(Gr(MIN:MAX2),x_L_c,'c')
        plot(Gr(MIN:MAX2),x_L_d,'c')
    end
    plot(Gr(MIN2:MAX3),x_L_a,'c')
    plot(Gr(MIN2:MAX3),x_L_b,'c')
    condition = Gr(MIN2)^2 + x_L_d(1)^2 > .985;
    if(X<=1)
        if(mod(X,.1)==0)
            if(mod(X,1)==0)
                word = [num2str(X) '.0'];
            else
                word = num2str(X);
            end
            if(X==1)
                angle = 90;
            else
                angle = -atan(angle_point)*180/pi;

set(text(Gr(MIN2),x_L_d(1),word),'Rotation',angle,'VerticalAlignment','bottom','HorizontalAlign
ment','left');

set(text(Gr(MIN2),-x_L_d(1),word),'Rotation',-angle+180,'HorizontalAlignment','right','VerticalAl
ignment','bottom');
            end
            if(mod(X,.2)==0)
            xval = X^2/(X^2+4);
            yval = 1/2/X*(-2+2*(1-X^2+2*X^2*xval-X^2*xval^2)^(1/2));
            angle = -atan(yval/(.5-xval))*180/pi;

set(text(xval,yval,word),'Rotation',angle,'HorizontalAlignment','left','VerticalAlignment','bottom');

set(text(xval,-yval,word),'Rotation',-angle+180,'HorizontalAlignment','right','VerticalAlignment','b
ottom')
            end
        end
    elseif(X<=2)
        if(mod(X,.2)==0)
            if(mod(X,1)==0)
                word = [num2str(X) '.0'];
            else
                word = num2str(X);
            end
            if(condition)
                angle = -atan(angle_point)*180/pi+180;

set(text(Gr(MIN2),x_L_a(1),word),'Rotation',angle,'VerticalAlignment','bottom','HorizontalAlign
m ent','left');

set(text(Gr(MIN2),-x_L_a(1),word),'Rotation',-angle+180,'HorizontalAlignment','right','VerticalAl
i gnment','bottom');
            else
                angle = -atan(angle_point)*180/pi+180;

set(text(Gr(MAX2),x_L_d(length(x_L_d)),word),'Rotation',angle,'VerticalAlignment','botto
m','HorizontalAlignment','left');

set(text(Gr(MAX2),-x_L_d(length(x_L_d)),word),'Rotation',-angle+180,'HorizontalAlignment' ,'ri
ght','VerticalAlignment','bottom');
            end
        end
    elseif(X<=5)
        if(mod(X,1)==0)
            if(condition)
                angle = -atan(angle_point)*180/pi+180;
                set(text(Gr(MIN2),x_L_a(1),[num2str(X)
'.0']),'Rotation',angle,'VerticalAlignment','botto m','HorizontalAlignment','left');
            set(text(Gr(MIN2),-x_L_a(1),[num2str(X)
'.0']),'Rotation',-angle+180,'HorizontalAlignment ','right','VerticalAlignment','bottom');
            else
                angle = -atan(angle_point)*180/pi+180;
                set(text(Gr(MAX2),x_L_d(length(x_L_d)),[num2str(X)
'.0']),'Rotation',angle,'VerticalAlignment','bottom','HorizontalAlignment','left');
                set(text(Gr(MAX2),-x_L_d(length(x_L_d)),[num2str(X)
'.0']),'Rotation',-angle+180,'HorizontalAlignment','right','VerticalAlignment','bottom');
            end
        end
    else
        if(mod(X,10)==0)
            if(condition)
                angle = -atan(angle_point)*180/pi+180;

set(text(Gr(MIN2),x_L_a(1),num2str(X)),'Rotation',angle,'VerticalAlignment','bottom','Horizontal
Alignment','left');

set(text(Gr(MIN2),-x_L_a(1),num2str(X)),'Rotation',-angle+180,'HorizontalAlignment','right','Ver
ticalAlignment','bottom');
            else
                angle = -atan(angle_point)*180/pi+180;

set(text(Gr(MAX2),x_L_d(length(x_L_d)),num2str(X)),'Rotation',angle,'VerticalAlignment','botto
m','HorizontalAlignment','left');

set(text(Gr(MAX2),-x_L_d(length(x_L_d)),num2str(X)),'Rotation',-angle+180,'HorizontalAlignm
ent','right','VerticalAlignment','bottom');
            end
        end
    end
end
%plot imaginary axis
plot(zeros(1,length(Gr)),Gr,'r');
wavelengths = 0:.01:.5;
angle = linspace(pi,-pi,length(wavelengths));
```

```
wave_circle = 1.05*exp(1i*phaseAngle);
plot(real(wave_circle),imag(wave_circle),'r');
for i=1:length(wavelengths)-1
        x=real(1.025*exp(1i*angle(i)));
        y=imag(1.025*exp(1i*angle(i)));
        if(x>0)
                rot_angle=atan(y/x)*180/pi-90;
        else
                rot_angle=atan(y/x)*180/pi+90;
        end
        if(wavelengths(i)==0)
                word = '0.00';
        elseif(mod(wavelengths(i),.1)==0)
                word = [num2str(wavelengths(i)) '0'];
        else
                word = num2str(wavelengths(i));
        end

set(text(x,y,word),'Rotation',rot_angle,'VerticalAlignment','middle','HorizontalAlignment','center')
end

% --- Executes on button press in pushbutton3.

function edit1_Callback(hObject, eventdata, handles)
% hObject        handle to edit1 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles        structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%               str2double(get(hObject,'String')) returns contents of edit1 as a double
  zl = str2double(get(hObject,'String'));
        if isnan(zl)
                errordlg('You must input a number.');
                set(handles.edit1, 'string', '');
        end

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject        handle to edit1 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles        empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%             See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
end

function edit2_Callback(hObject, eventdata, handles)
% hObject        handle to edit2 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles        structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%               str2double(get(hObject,'String')) returns contents of edit2 as a double
  zl = str2double(get(hObject,'String'));
        if isnan(zl)
                errordlg('You must input a number.');
                set(handles.edit2, 'string', '');
        end
```

```
% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a double

% --- Executes during object creation, after setting all properties.

function edit9_Callback(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit9 as text
%        str2double(get(hObject,'String')) returns contents of edit9 as a double
zl = str2double(get(hObject,'String'));
    if isnan(zl)
        errordlg('You must input a number.');
        set(handles.edit9, 'string', '');
    elseif real(zl<0)
        errordlg('The real part cannot be negative.');
        set(handles.edit9, 'string', '');
    end

% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit10_Callback(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
  zl = str2double(get(hObject,'String'));
    if isnan(zl)
        errordlg('You must input a number.');
        set(handles.edit10, 'string', '');
    end
% Hints: get(hObject,'String') returns contents of edit10 as text
%        str2double(get(hObject,'String')) returns contents of edit10 as a double
% --- Executes during object creation, after setting all properties.

function edit10_CreateFcn(hObject, eventdata, handles)
% hObject        handle to edit10 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles        empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%              See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject        handle to pushbutton4 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles        structure with handles and user data (see GUIDATA)
zl = str2double(get(handles.edit9,'string'));
z2 = str2double(get(handles.edit10,'string'));
z3 = str2double(get(handles.edit2,'string'));
d = str2double(get(handles.edit1,'string'));
z = ((zl+j*z2)./z3+j*tan(2*pi*d))/(1+j*((zl+j*z2)./z3)*tan(2*pi*d));
z22=z*z3;
y = 1/z22;
aida = (z-1)/(z+1);
set(handles.text19,'string',num2str(z22))
set(handles.text21,'string',num2str(y))
set(handles.text24,'string',num2str(aida))
p=str2double(get(handles.text24,'string'));
p1=real(p);
p2=imag(p);
p3=sqrt(p1^2+p2^2);
set(handles.text26,'string',num2str(p3))
r = str2double(get(handles.text24,'string'));
r1 = real(r);
r2 = imag(r);
r3 = sqrt((r1)^2+(r2)^2);
r4 = (1+r3)./(1-r3);
set(handles.text23,'string',r4)
%画等反射系数圆
zl = str2double(get(handles.edit9,'string'));
R=zl./z3;
X=z2./z3;
U=(R^2+X^2-1)./(R^2+2*R+1+X^2);
V=(2*X)./(R^2+2*R+1+X^2);
tr = 2*pi*(0:0.01:1);
r=sqrt((R^2-2*R+1+X^2)/(R^2+2*R+1+X^2));
plot(r*cos(tr),r*sin(tr),'r','LineWidth',1)
pause(0.5)
%  等归一化电抗圆
for rr = 1/(1+R)
        cr = 1-rr;
        plot(cr+rr*cos(tr),rr*sin( tr),'r','LineWidth',1)
end
pause(0.5)
for x=X
        rx= 1/x;
        cx= rx;
        tx=2* atan( x) * ( 0:0.01:1);
        if tx<pi
                plot(1-rx*sin(tx),cx-rx*cos(tx),'r','LineWidth',1)
        else
                plot(1-rx*sin(tx),-cx+rx*cos(tx),'r','LineWidth',1)
        end
end
pause(0.5)
z11 = ((zl+j*z2)./z3+j*tan(2*pi*0))/(1+j*((zl+j*z2)./z3)*tan(2*pi*0));
aida11 = (z11-1)/(z11+1);
plot([0 4*real(aida11)],[0 4*imag(aida11)],'r','LineWidth',1)
text(real(aida11),imag(aida11),['(',num2str(real(aida11)),',',num2str(imag(aida11)),')'],'color','b','fo
ntsize',9);
hold on
pause(0.5)
z = ((zl+j*z2)./z3+j*tan(2*pi*d))/(1+j*((zl+j*z2)./z3)*tan(2*pi*d));
aida = (z-1)/(z+1);
plot([-4*real(aida) 4*real(aida)],[-4*imag(aida) 4*imag(aida)],'r','LineWidth',1)
text(real(aida),imag(aida),['(',num2str(real(aida)),',',num2str(imag(aida)),')'],'color','b','fontsize',9);
text(-real(aida),-imag(aida),['(',num2str(-real(aida)),',',num2str(-imag(aida)),')'],'color','b','fontsize',
9);


% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.text21,'string',")
set(handles.text19,'string',")
set(handles.text23,'string',")
set(handles.text24,'string',")
set(handles.text26,'string',")
set(handles.edit9,'string',")
set(handles.edit10,'string',")
set(handles.edit1,'string',")
set(handles.edit2,'string',")
axes(handles.axes1);
cla reset %清空图像

% --- Executes during object creation, after setting all properties.
function text24_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text24 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% --- Executes during object creation, after setting all properties.
function text23_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text23 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% --- Executes during object creation, after setting all properties.
function text21_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text21 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% --- Executes during object creation, after setting all properties.
function text22_CreateFcn(hObject, eventdata, handles)
```

```
% hObject       handle to text22 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles       empty - handles not created until after all CreateFcns called

% --- Executes during object creation, after setting all properties.
function text19_CreateFcn(hObject, eventdata, handles)
% hObject       handle to text19 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles       empty - handles not created until after all CreateFcns called
%
% --- Executes during object creation, after setting all properties.
function text20_CreateFcn(hObject, eventdata, handles)
% hObject       handle to text20 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles       empty - handles not created until after all CreateFcns called


% --- Executes during object creation, after setting all properties.
function text26_CreateFcn(hObject, eventdata, handles)
% hObject       handle to text26 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles       empty - handles not created until after all CreateFcns called
```

第六部分：参考文献

【1】David M.Pozar. 《微波工程》第三版 电子工业出版社，2015

【2】百度百科—— GUI

【3】《基于 matlab 的 smith 圆图的设计》 百度文库