基于 SMITH 圆图工具的研究与开发

——《微波技术与天线》大作业

目录

1 SMITH 圆图历史背景

Smith 圆图是射频和微波工程领域中非常重要的工具，它以简单的图形表示了阻抗和导纳的等效电路元件。通过使用 Smith 圆图，电路设计师可以方便地进行电路分析和设计。

Smith 圆图是由美国工程师 Oscar K. Smith 于 1936 年发明的。当时，射频和微波技术正在快速发展，对于复杂的阻抗匹配网络设计的需求日益增长。Smith 圆图作为一种简便的阻抗匹配计算工具，迅速得到了广泛的应用和认可。Smith 圆图的发明得益于 Smith 在贝尔实验室的工作。贝尔实验室是一家在通信和电子工程领域享有盛誉的研究机构，其在 20 世纪二三十年代开展了大量的研究和开发工作。在这个时期，射频和微波技术得到了越来越多的关注和应用。在贝尔实验室工作期间，Smith 深刻地意识到阻抗匹配在通信和电子工程中的重要性。他发现，传统的阻抗计算方法不仅繁琐，而且容易出错。因此，他开始寻找一种更加简便的计算方法，最终发明了 Smith 圆图。

2 软件开发平台

Table 1: 软件开发平台及版本

名称	版本
MATLAB	R2021b
Qt 6	6.5.2
Qt Creator	11.0.2
MinGW	11.2.0

3 基于 MATLAB 的 SMITH CHART 程序开发与设计

3.1 程序总体框架

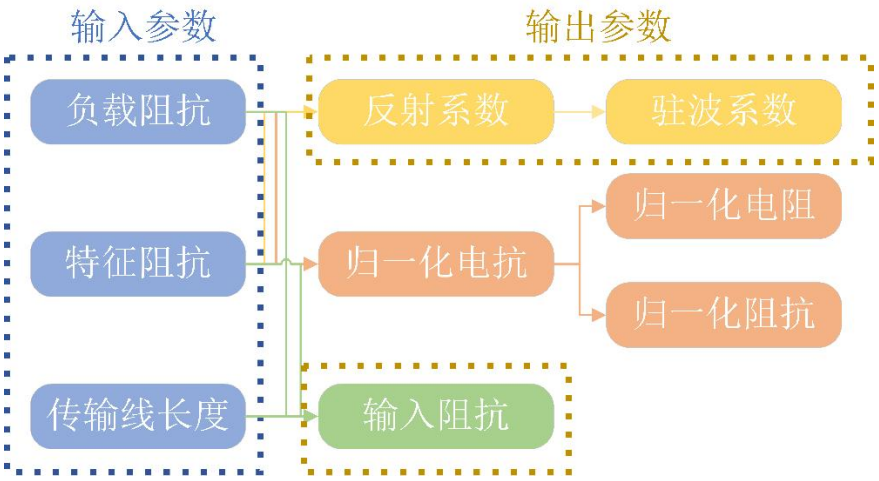

Figure 1 3 基于 MATLAB 的 SMITH CHART 程序框架

3.2 程序算法流程

3.2.1 特征参数归一化

归一化负载阻抗：

$$\bar{Z}_l = \frac{Z_l}{Z_0} \tag{1}$$

归一化传输线长度：

$$\bar{L}_0 = \frac{L_0}{\lambda} \tag{2}$$

3.2.2 反射系数与驻波系数计算

反射系数：

$$\Gamma_l = \frac{Z_l - Z_0}{Z_l + Z_0} \tag{1}$$

驻波系数：

$$\rho = \frac{1+|\Gamma_l|}{1-|\Gamma_l|} \tag{2}$$

3.2.3 归一化电阻圆

归一化电阻圆方程：

$$\left(\Gamma_u - \frac{r}{1+r}\right)^2 + \Gamma_v^{\,2} = \left(\frac{1}{1+r}\right)^2 \tag{1}$$

将其转换到极坐标系：

$$x_1 = \frac{r}{1+r} + \frac{1}{1+r} \times \cos(\theta) \tag{2}$$

$$y_1 = \frac{1}{1+r} \times \sin(\theta) \tag{3}$$

3.2.4 归一化电抗圆

归一化电抗圆方程：

$$\left(\Gamma_u - 1\right)^2 + \left(\Gamma_v - \frac{1}{x}\right)^2 = \left(\frac{1}{x}\right)^2 \tag{1}$$

将其转换到极坐标系：

$$x_2 = 1 + \frac{1}{x} \times \cos(\theta) \tag{2}$$

$$y_2 = \frac{1}{x} + \frac{1}{x} \times \sin(\theta) \tag{3}$$

3.2.5 输入阻抗计算

$$\mathrm{Z_i} = \mathrm{Z_0}\,\frac{\mathrm{Z}_l + j\mathrm{Z}_0 \tan(\bar{L}_0 \times 2\pi)}{\mathrm{Z}_0 + j\mathrm{Z}_l \tan(\bar{L}_0 \times 2\pi)} \tag{1}$$

3.3 程序功能测试

Table 2 输入输出数据测试表 1

输入数据说明	输入数据参数	输出数据说明	输出数据参数
负载阻抗 Z_l	100+50j	反射系数	0.4000+0.2000i
特征阻抗 Z_0	50	驻波系数	2.6180
传输线长度 L_0	0.34λ	输入阻抗	21.0244+14.5475i

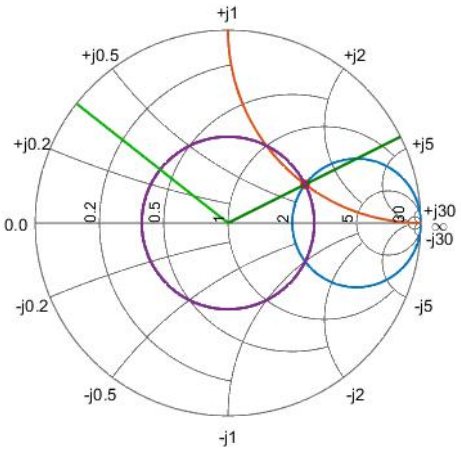

Figure 2 输出 SMITH 圆测试图 1

Table 3 输入输出数据测试表 2

输入数据说明	输入数据参数	输出数据说明	输出数据参数
负载阻抗 Z_l	50-100j	反射系数	0.5000-0.5000i
特征阻抗 Z_0	50	驻波系数	5.8284
传输线长度 L_0	0.4λ	输入阻抗	1.0420e+02+1.3380e+02i

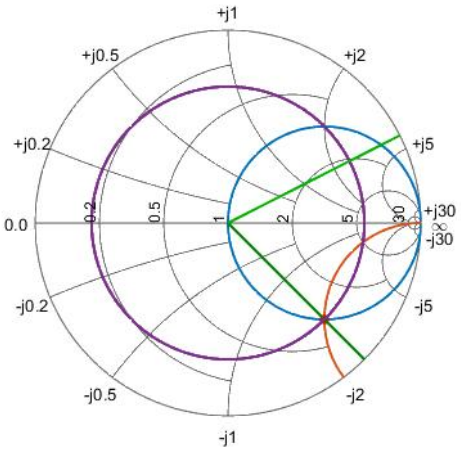

Figure 3 输出 SMITH 圆测试图 2

如 Figure 2、Figure 3 所示，紫色是以反射系数的模值为半径，原点为圆心的圆；蓝色为归一化电阻圆图；橙色为归一化电抗圆图；深绿色为归一化负载阻抗线，浅绿色为归一化输入阻抗线，从负载到电源输入为顺时针方向旋转。

4 基于 QT6 C++的 SMITH CHART 软件开发与设计

Matlab 提供了丰富的工具箱，可以快速原型设计各种数学、科学和工程应用程序，具有大量的内置函数和绘图工具，方便进行数据分析和可视化。

但是，Matlab 是闭源的，用户无法查看和修改底层代码，而 Qt 是开源的，可以免费使用，并提供了商业许可证选项。这使得它在开发成本方面更具竞争力。此外，Qt 使用 C++编程语言，具有高性能，适用于需要处理大数据或计算密集型任务的应用程序。

所以，此处选用 QT6 C++进行 SMITH CHART 软件开发与设计。

4.1 软件 gui 界面

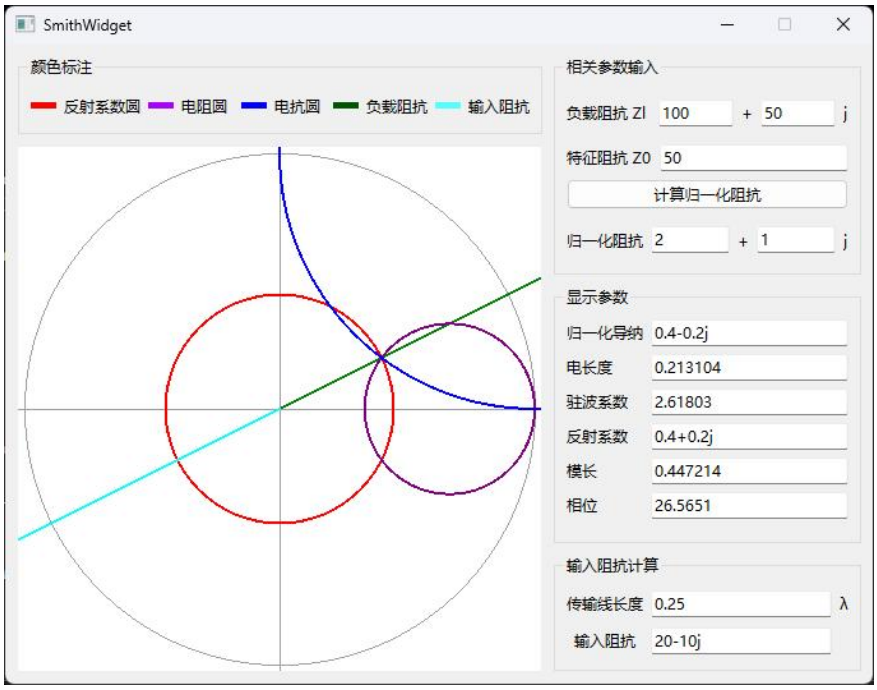

Figure 4 软件 gui 界面图

4.2 软件功能介绍

通过 QPainter 使用事件滤波器在 widget_paint 进行图像绘制，输入相关参数，每隔 0.5 秒实时更新绘图和参数。

输入负载阻抗和特征阻抗可计算归一化阻抗，也可直接输入归一化阻抗进行后续的绘图与计算。

4.3 软件功能测试

测试 1 Figure 5 与 Table 2 结果一致：

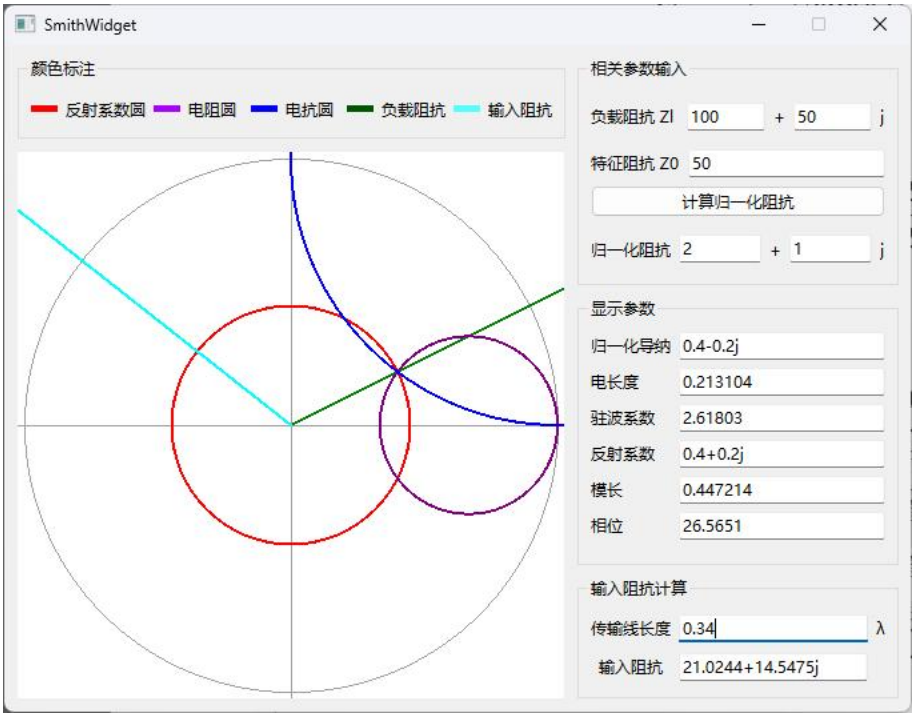

Figure 5 SMITH 软件测试 1 结果图

测试 2 Figure 6 与 Table 3 结果一致：

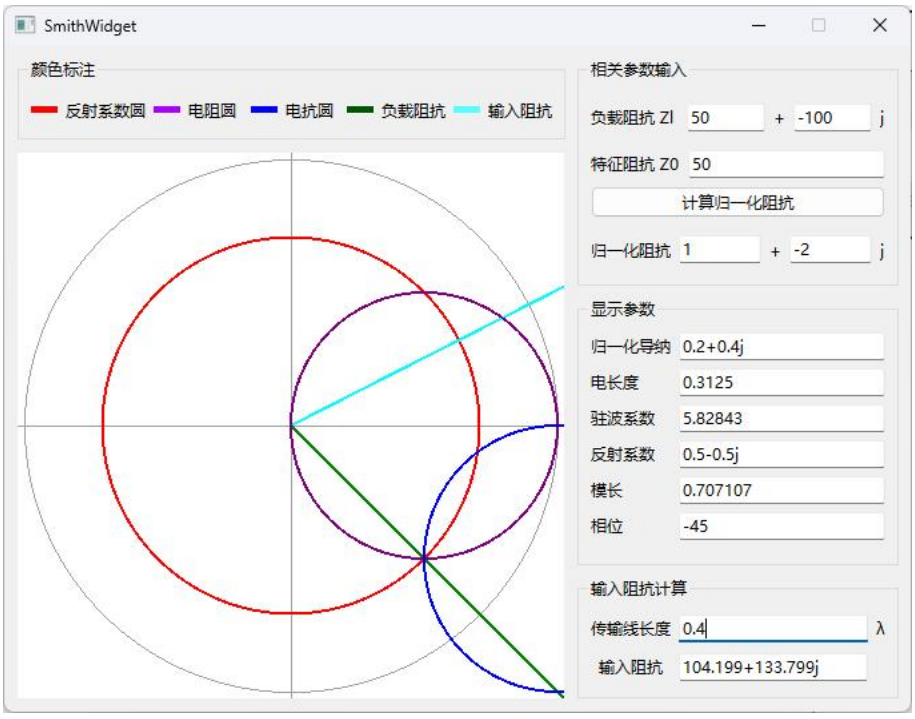

Figure 6 SMITH 软件测试 2 结果图

5 总结与展望

通过本次作业，我深入了解了如何使用 Matlab 和 Qt6 C++来解决实际问题，这有助于我将 Smith Chart 理论知识应用于实践。熟练掌握这两种工具的使用，提高了我的计算和编程能力，也拓宽了我的工程和科研视野。在学习过程中，不仅要学习工具的使用，还要理解背后的原理和算法，这有助于更好地应对复杂的问题。

总的来说，通过本次学习，我获得了有关 Matlab、Qt6 C++的宝贵经验，这将对我的未来学术和职业发展产生积极影响。我期待将这些知识和技能应用到更多的项目中，继续提升自己的能力。

参考文献

[1] Shamim A, Radwan A G, Salama K N. Fractional smith chart theory[J]. IEEE Microwave and Wireless Components Letters, 2011, 21(3): 117-119.

[2] Pereira J R, Pinho P. Using modern tools to explain the use of the Smith chart[J]. IEEE Antennas and Propagation Magazine, 2010, 52(2): 145-150.

[3] Chan K C, Harter A. Impedance matching and the Smith chart-The fundamentals[J]. RF DESIGN, 2000, 23(7): 52-66.

[4] Sherriff N. Learn Qt 5: Build modern, responsive cross-platform desktop applications with Qt, C++, and QML[M]. Packt Publishing Ltd, 2018.

[5] 房少军，王昊. Smith 圆图教学软件的开发[J]. 实验室研究与探索, 2012 (3): 51-54.

[6] 王国辉. 论史密斯圆图的美[J]. 美与时代: 创意 (上), 2017 (8): 22-24.

[7] 戢予，徐庆思，陈园园，等. 史密斯圆图的原理分析及应用[J]. 数码世界, 2017 (8): 19-20.

[8] Plot measurement data on a Smith chart - MATLAB smithplot (mathworks.com)

[9] 微波课设_基于 pyqt5+pyqtgraph 的 Smith 圆图 GUI 程序设计_python 画史密斯图-CSDN 博客

[10] Smith 圆图教学课件使用说明 - 豆丁网 (docin.com)

[11] Hanfu-Zhang/Smith (github.com)

附录

附录 1

介绍：MATLAB 程序代码

```
%%*********************************************************************
        % @File Name: Smith.m
        % @Author: Accelerator(Xu HuiYao)
        % @Version: 2.0
        % @Mail: 2364412203@qq.com
%% 清空
clear; clc; close;
%% 输入参数
Zl_str = input('请输入负载阻抗:');
Re = regexp(Zl_str, '+', 'split');
if(Re==Zl_str)
    Re = regexp(Zl_str, '-', 'split');
    Im = regexp(char(Re(2)), 'j', 'split');
    Zl = str2num(char(Re(1)))-j*str2num(char(Im(1)));
else
    Im = regexp(char(Re(2)), 'j', 'split');
    Zl = str2num(char(Re(1)))+j*str2num(char(Im(1)));
end
Z0 = input('请输入特征阻抗:');
L0 = input('请输入传输线长度λ:');
%% 计算归一化阻抗
Z_normalization = Zl/Z0;
%% 计算归一化电阻 r、归一化电抗 x
r = real(Z_normalization);
```

```
x = imag(Z_normalization);
%% 计算反射系数
Gamma = (Zl - Z0) / (Zl + Z0);
%% 计算驻波比
VSWR = (1+abs(Gamma))/(1-abs(Gamma));
%% 创建 Smith 图
figure;
set(gcf, 'Name', 'Smith Chart'); % 修改 figure 窗口的标题
smithplot;
hold on;
%% 绘制归一化电阻圆
theta = 0: pi/100: 2*pi;
C_r = r / (1 + r); % 中心坐标(C_r,0)
R_r = 1 / (1 + r); % 电阻圆半径
% 转换到极坐标
x_r = C_r + R_r * cos(theta);
y_r = R_r * sin(theta);
plot(x_r, y_r, 'LineWidth', 1.5);
hold on;
%% 绘制归一化电抗圆
theta = 0: pi/100: 2*pi;
C_x = 1 / x;
R_x = 1 / x;
x_x = 1 + R_x * cos(theta);
y_x = C_x + R_x * sin(theta);
plot(x_x, y_x,'LineWidth', 1.5);
hold on;
```

```
%% 绘制反射系圆
plot(Gamma, 'ro', 'MarkerSize', 5, 'MarkerFaceColor', 'r');
% 缩放反射系数，使其模值为1
normalized_Gamma = Gamma / abs(Gamma);
line([0, real(normalized_Gamma)], [0, imag(normalized_Gamma)],
'Color', [0, 0.5, 0], 'LineWidth', 1.5);
Gamma_abs = abs(Gamma); % 反射系数的模值
theta = linspace(0, 2*pi, 1000); % 反射系数圆角度为360°
x_circle = Gamma_abs * cos(theta);
y_circle = Gamma_abs * sin(theta);
plot(x_circle, y_circle, 'LineWidth', 2);
%% 计算输入阻抗
Zi_normalization = (Zl+j*Z0*tan(L0*2*pi))/(Z0+j*Zl*tan(L0*2*pi));
Zi = Zi_normalization*Z0;
Gamma_i = (Zi - Z0) / (Zi + Z0);
normalized_Gamma_i = Gamma_i / abs(Gamma_i);
line([0, real(normalized_Gamma_i)], [0, imag(normalized_Gamma_i)],
'Color', [0, 0.7, 0], 'LineWidth', 1.5);
%% 显示数据
hold off;
disp("----------计算结果如下----------");
disp('反射系数：');
disp(Gamma);
disp('驻波系数：');
disp(VSWR);
disp('输入阻抗：');
disp(Zi);
```

附录 2

介绍：Qt6 C++程序代码 smithwidget.cpp

```
/*****************************************************************
        * @File Name: smithwidget.c
        * @Author: Accelerator(Xu HuiYao)
        * @Version: 1.0
        * @Mail: 2364412203@qq.com
*****************************************************************/

#include "smithwidget.h"
#include <QDebug>
#include <QPainter>
#include "ui_smithwidget.h"
#include <cmath>

SmithWidget::SmithWidget(QWidget *parent)
    : QWidget(parent)
    , ui(new Ui::SmithWidget)
{
    ui->setupUi(this);
    //安装事件滤波器
    timer = new QTimer(this);
    connect(timer, &QTimer::timeout, this,
&SmithWidget::onTimerTimeout);
    timer->start(500); // 0.5 秒触发一次定时器

    ui->widget_paint->installEventFilter(this);
}

SmithWidget::~SmithWidget()
{
    delete ui;
}

bool SmithWidget::eventFilter(QObject *watched, QEvent *event)
{
    if (watched == ui->widget_paint && event->type() == QEvent::Paint)
{
        SmithPaint(); //响应函数
    }
```

```
    return QWidget::eventFilter(watched, event);
}

//实现响应函数
void SmithWidget::SmithPaint()
{
    QPainter painter(ui->widget_paint);
    painter.setPen(Qt::gray);
    //绘制单位圆
    painter.drawEllipse(QPoint(ui->widget_paint->width() / 2,
ui->widget_paint->height() / 2),
                        ui->widget_paint->width() / 2 - PAINT_BIAS,
                        ui->widget_paint->height() / 2 - PAINT_BIAS);
    //比例系数
    double k = (ui->widget_paint->width() / 2 - PAINT_BIAS);

    //绘制横轴
    painter.drawLine(QPoint(ui->widget_paint->width(),
ui->widget_paint->height() / 2),
                     QPoint(0, ui->widget_paint->height() / 2));
    //绘制纵轴
    painter.drawLine(QPoint(ui->widget_paint->width() / 2, 0),
                     QPoint(ui->widget_paint->width() / 2,
ui->widget_paint->height()));

    //归一化阻抗
    z = std::complex<double>(ui->lineEdit_z_r->text().toDouble(),
                             ui->lineEdit_z_i->text().toDouble());

    //归一化导纳
    y = 1.0 / z;
    if (y.imag() >= 0)
        ui->lineEdit_y->setText(QString::number(y.real()) + "+" +
QString::number(y.imag()) + "j");
    else if (y.imag() < 0)
        ui->lineEdit_y->setText(QString::number(y.real()) +
QString::number(y.imag()) + "j");

    zl = std::complex<double>(ui->lineEdit_zl_r->text().toDouble(),
                              ui->lineEdit_zl_i->text().toDouble());
    z0 = ui->lineEdit_z0->text().toDouble();
```

```
    //反射系数
    gamma = (zl - z0) / (zl + z0);

    if (gamma.imag() >= 0)
        ui->lineEdit_gamma->setText(QString::number(gamma.real()) + "+"
                                    + QString::number(gamma.imag()) + "j");
    else if (gamma.imag() < 0)
        ui->lineEdit_gamma->setText(QString::number(gamma.real()) +
QString::number(gamma.imag())
                                    + "j");

    //模长
    ui->lineEdit_abs->setText(QString::number(abs(gamma)));
    //相位
    ui->lineEdit_angle->setText(QString::number(arg(gamma) * 180 /
3.1415926));

    //驻波系数
    vswr = (1 + std::abs(gamma)) / (1 - std::abs(gamma));
    ui->lineEdit_vswr->setText(QString::number(vswr));

    //电长度
    ui->lineEdit_l->setText(QString::number(0.25 - (arg(gamma) / 4 /
3.1415926)));

    //输入阻抗
    l0 = ui->lineEdit_l0->text().toDouble();
    zi = z0 * (zl + std::complex<double>(0, 1) * z0 * tan(l0 * 2 * 3.1415926))
         / (z0 + std::complex<double>(0, 1) * zl * tan(l0 * 2 * 3.1415926));

    if (zi.imag() >= 0)
        ui->lineEdit_zi->setText(QString::number(zi.real()) + "+" +
QString::number(zi.imag())
                                 + "j");
    else if (zi.imag() < 0)
        ui->lineEdit_zi->setText(QString::number(zi.real()) +
QString::number(zi.imag()) + "j");

    gamma_i = (zi - z0) / (zi + z0);

    //反射系数圆
    painter.setPen(QPen(Qt::red, 2));
```

```
    painter.translate(ui->widget_paint->width() / 2,
                      ui->widget_paint->height() / 2); // 将坐标原点移到窗口
中心
    painter.drawEllipse(QPointF(0, 0), abs(gamma) * k, abs(gamma) * k);

    painter.setPen(QPen(Qt::darkGreen, 2));
    painter.drawLine(QPointF(0, 0), QPointF(gamma.real() * k * 500,
-gamma.imag() * k * 500));

    painter.setPen(QPen(Qt::cyan, 2));
    painter.drawLine(QPointF(0, 0), QPointF(gamma_i.real() * k * 500,
-gamma_i.imag() * k * 500));

    //归一化电阻圆
    painter.setPen(QPen(Qt::darkMagenta, 2));
    painter.drawEllipse(QPointF(z.real() / (1 + z.real()) * k, 0),
                        1 / (1 + z.real()) * k,
                        1 / (1 + z.real()) * k);

    //归一化电抗圆
    painter.setPen(QPen(Qt::blue, 2));
    painter.drawEllipse(QPointF(1 * k, -1 / z.imag() * k), 1 / z.imag()
* k, 1 / z.imag() * k);
}

//按键计算归一化阻抗
void SmithWidget::on_pushButton_clicked()
{
    z_r = ui->lineEdit_zl_r->text().toDouble() /
ui->lineEdit_z0->text().toDouble();
    ui->lineEdit_z_r->setText(QString::number(z_r));
    z_i = ui->lineEdit_zl_i->text().toDouble() /
ui->lineEdit_z0->text().toDouble();
    ui->lineEdit_z_i->setText(QString::number(z_i));
}
```