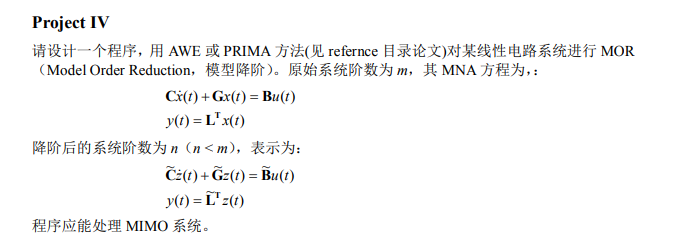


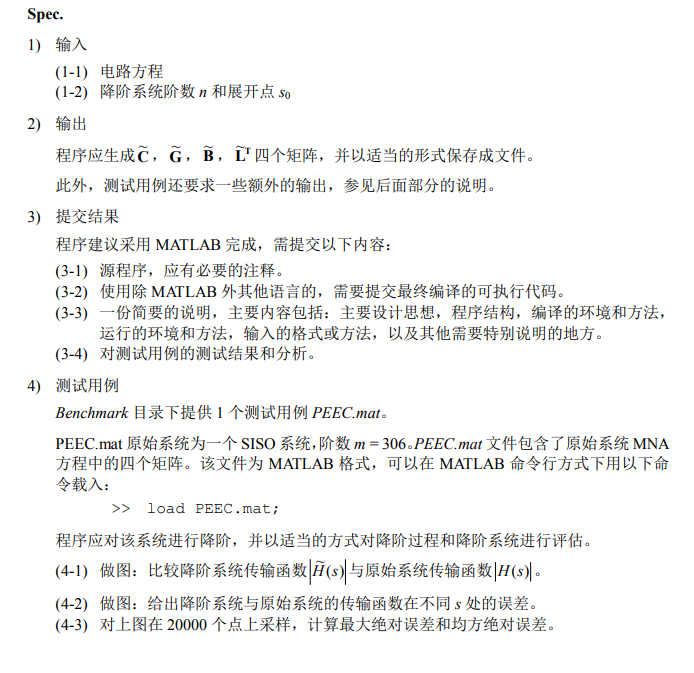
**课程报告**

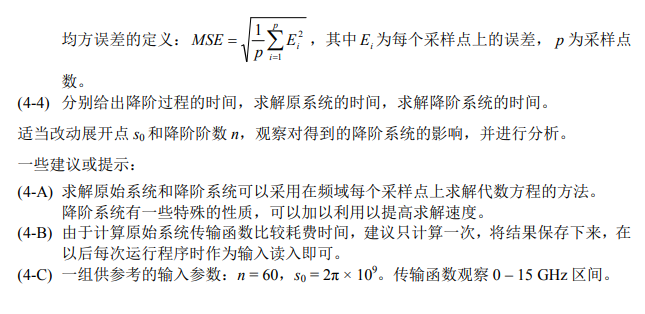
|  |  |
| --- | --- |
| **课程名称：现代集成电路分析方法** | **Project： project8** |
| **姓 名： 王凯旋** | **学 号： 21112020061** |
| **学 院： 微电子学院** | **专 业： 集成电路与系统分析** |

# 1.设计要求



# 2.设计标准





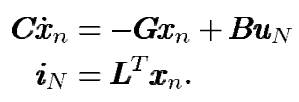
# 3.设计思路

使用PRIMA的方法进行方程组降阶。

由于规模非常大的线性方程组求解需要花费大量的时间和计算资源，所以对需要求解的系统进行降解是一个有效且必要的手段。PRIMA是一种基于krylov子空间方法的讲解方法，通过快Arnoldi算法发展而来。该方法不仅能够对任意的RLC电路进行降阶还可以保持电路的稳定性和无源性，从而保证其稳定性。除此之外，PRIMA还能用于MIMO电路模型的降阶。

在这里我们给出论文中的以下定义

若有MNA方程组如下



则可以通过如下变换



得到



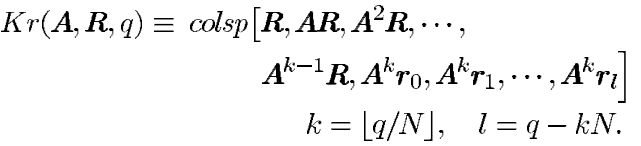
在这里我们定义块矩



其中



同时定义块Krylov空间



通过PRIMA.pdf中介绍的算法编写出如下程序

# 4. 程序构成

计算所需要的函数如下表所示

|  |  |
| --- | --- |
| Main.m | 读入已经计算好的原始系统传输函数和所需时间，并读入PEEC.mat所保存的SISO系统，调用PRIMA进行处理并进行降阶后求解。 |
| Ori\_sol.m | 以求逆的方式，直接求解原始系统的传递函数。 |
| PRIMA.m | 参照论文的方法对系统进行降阶，可以选择阶数和展开点 |
| Sol\_after\_PRIMA.m | 降阶结束对降阶后系统进行求解 |

# 5. 程序流程

程序的执行如下：

1. 读入已经计算好的原始系统传输函数和所需时间。
2. 使用PRIMA方法对系统进行降阶，计算降阶所需时间
3. 降阶结束后对降阶后系统求解，计算所需时间
4. 误差分析

PRIMA降阶算法以伪代码的方式给出如下所示

由MNA方程组得到系数矩阵C，G ，B，LT

求解G\*R=B 得到R

对R进行qr分解，得到X0

令n=[q/N]+1

for k=1:1:需要计算的矩阵X的列数

令V=C\*Xk-1

求解G\*Xk(0)=V 得到Xk(0)

for j=1:1:k-1

令H =Xk-j’ \* Xk(j-1)

Xk(j) = Xk(j-1) – Xk-j\*H

endfor

对Xk(k)进行qr分解，得到Xk

endfor

令矩阵X为[X0 X1 …. Xk-1]的前q列

计算出Cq=X’\*C\*X , Gq=X’\*G\*X , Bq=Xq'\*B , LTq=Xq'\*L

# 6. 编译环境与运行方法

## 编译环境

本程序在win10 2021b环境下运行。

## 运行方法

更改main函数中所要更改的阶数和展开点

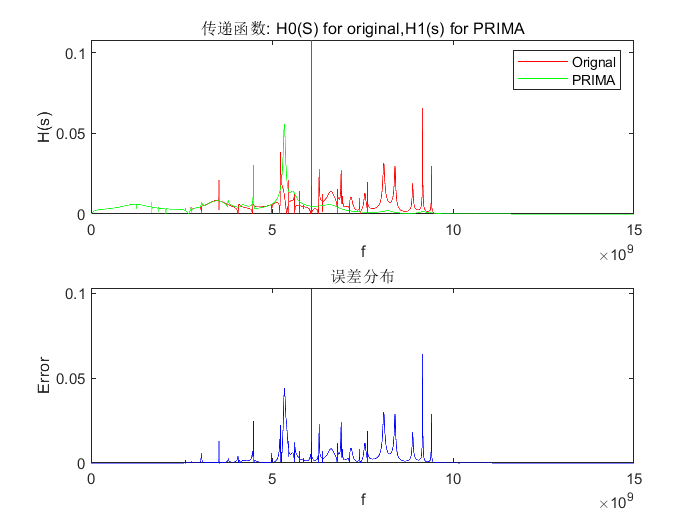
将工作目录放在代码目录下，执行main函数，便可输出相应结果

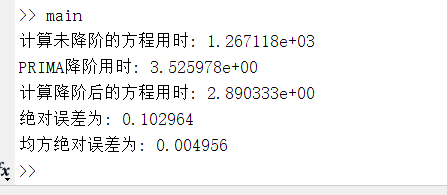
# 7. 输入输出格式

输入为PEEC.mat文件和原始方式执行得到的传输函数和所需时间，输出为传输函数比较的波形文件，以及经过降阶后得到的传输函数与原始方式得到的传输函数的误差

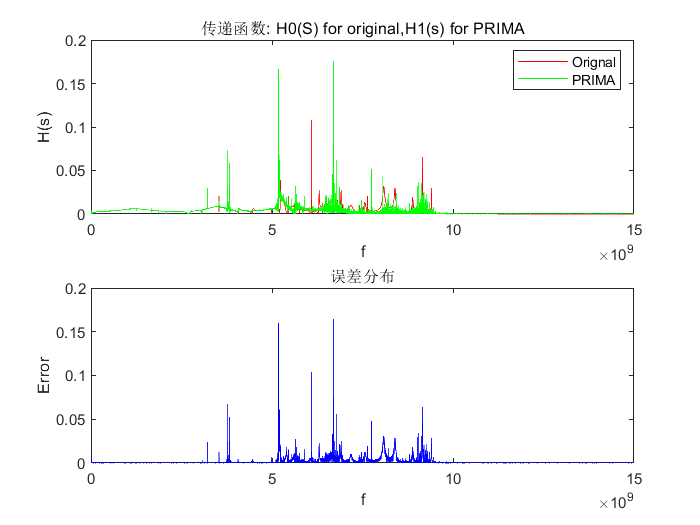
# 8. 测试结果和分析

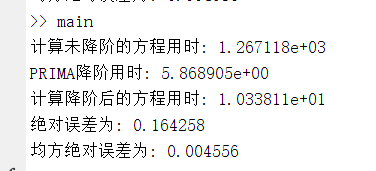
## 阶数=60; 展开点s0=2\*pi\*1e9



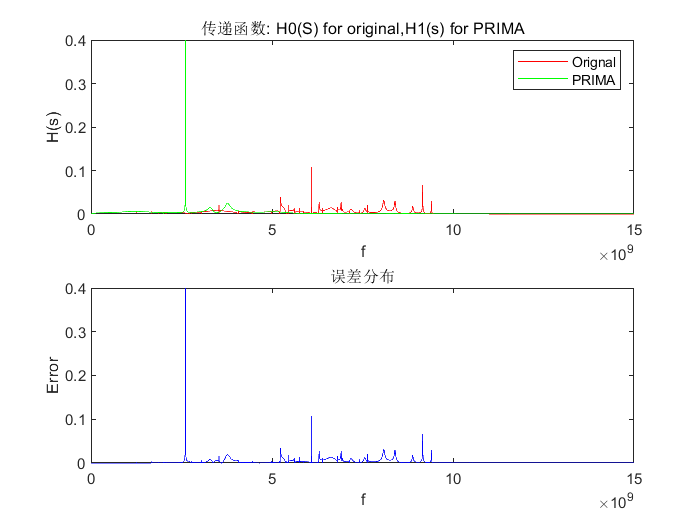


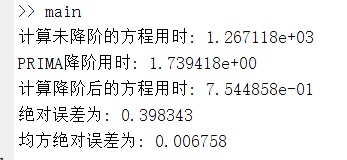
## 阶数=100;展开点=2\*pi\*1e9





## 阶数=30;展开点=2\*pi\*1e9





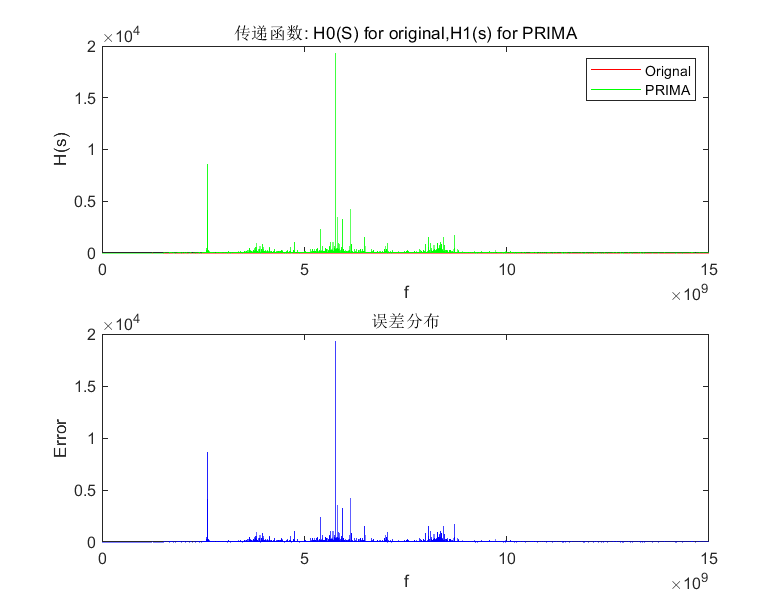
在展开点均为2pi\*1e9的情况下，得出如下表格

|  |  |  |  |
| --- | --- | --- | --- |
| 阶数 | 30 | 60 | 100 |
| 绝对误差 | 0.0398 | 0.1029 | 0.1462 |
| 均方绝对误差 | 0.0068 | 0.0050 | 0.0046 |
| PRIMA降阶用时 | 1.7394 | 3.5259 | 5.8689 |
| 计算降阶后方程用时 | 0.7545 | 2.8903 | 10.034 |

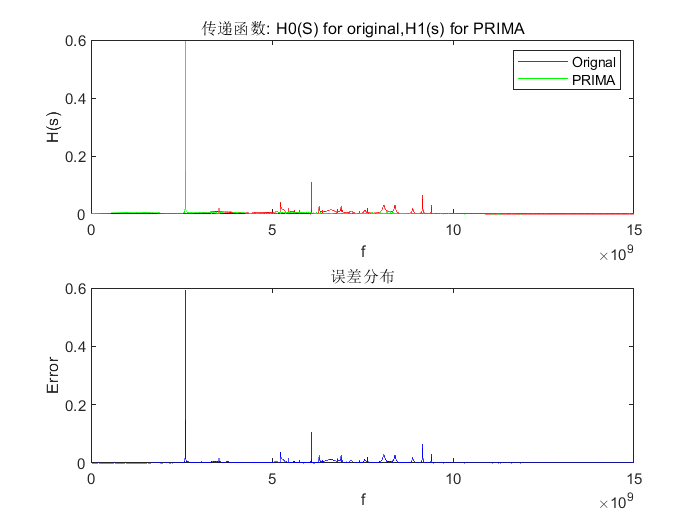
比较分析得到，降阶时，如果目标阶数越大，降阶用时越多，降阶用时线性增长；计算降阶后方程用时也越多，计算降阶后方程用时快速增长，但是均方绝对误差会变小；

接下来比较在展开阶数均为60的情况下，改变展开点的影响。

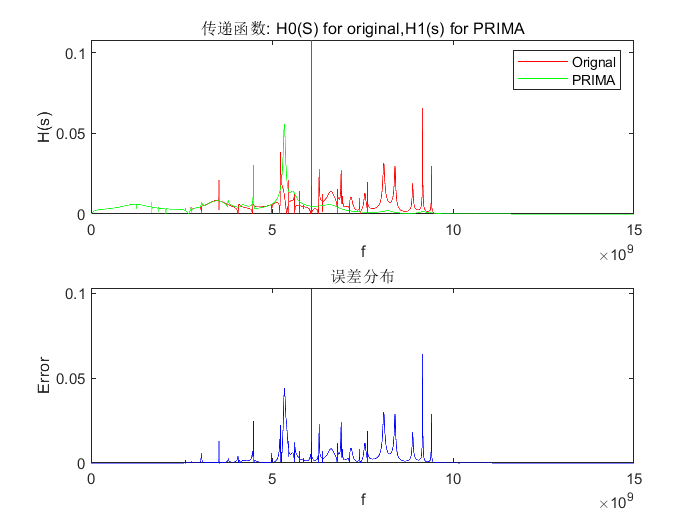
## 阶数=60;展开点=2\*pi\*1e7



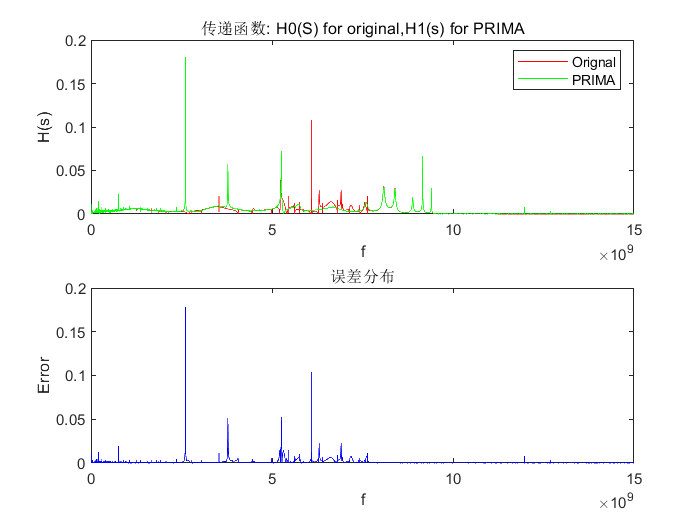
## 阶数=60;展开点=2\*pi\*1e8



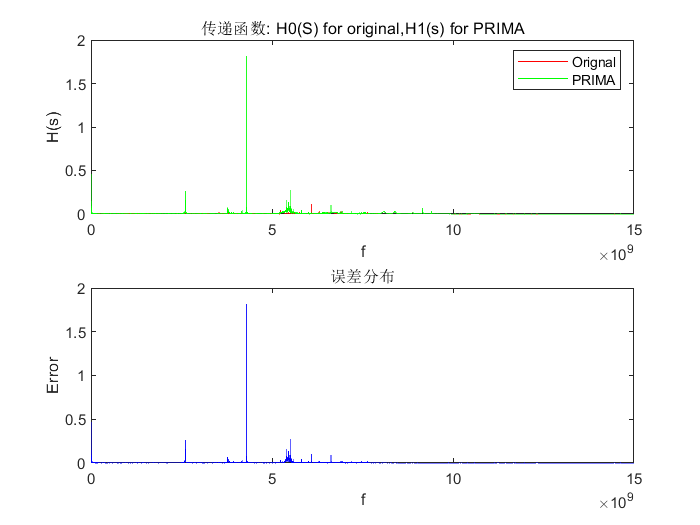
## 阶数=60;展开点=2\*pi\*1e9



## 阶数=60;展开点=2\*pi\*1e10



## 阶数=60;展开点=2\*pi\*1e11



通过比较上面五张图可以看出，在展开点附近，误差比较小，在远离展开点处，会有比较大的误差。

# 8. 总结

本次设计较好的完成了任务，通过project8可以使用PRIMA对高阶MNA方程进行降阶，节省计算所需时间，仿真结果误差在对应展开点处很小