1. 项目背景
2. 项目内容

本项目为控制系统设计与数字控制课程报告，目的是帮助我们从理论和实践两方面完成最小相位系统的设计与实现。最小相位系统的开环传递函数极点和零点的实部都小于或等于零，它也因而有着响应时间最小、能量延迟最小的特性，研究最小相位系统的设计，对于理解控制系统有着十分重要的意义。我们将在课程所学内容的基础上，先完成系统设计的理论推导，再选用Python或者MATLAB来进行仿真实践。

1. 设计指标

在已有被控对象性能指标的情况下，我们的目的是设计出可行性高、成本适中的控制器，从而使得整个控制系统的性能指标满足系统要求。常见的设计要求包括误差需求、相位裕量需求与截止频率需求，在给定被控对象的前提下，我们要尽可能设计出满足以上三个设计指标的控制器，从而构成性能佳、响应快、裕量大的最小相位系统。

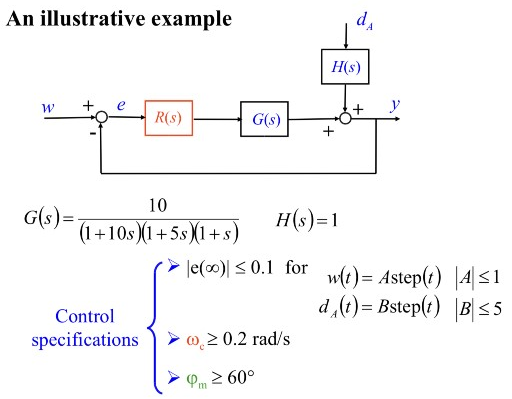
1. 理论推导

在本项目设计控制器的过程中，我们运用了三种不同的方案，在不同的设计指标下，三种方案有各自的优缺点，现先对三种方法进行理论上的推导。

* 1. case A

第一种方案，我们需要先根据系统的误差需求求出所设计控制器的系数项要求，再进一步画出常数项和被控对象所构成系统的波特图，与其余设计指标进行对比，如果满足指标则设计完成，如果不满足指标，我们需要调整控制器的零、极点，在系统的截止频率与设计指标相同的情况下，求出对应的数据要求，完成控制器的设计。

例如，当我们得到有如下被控对象的响应函数与设计指标时：



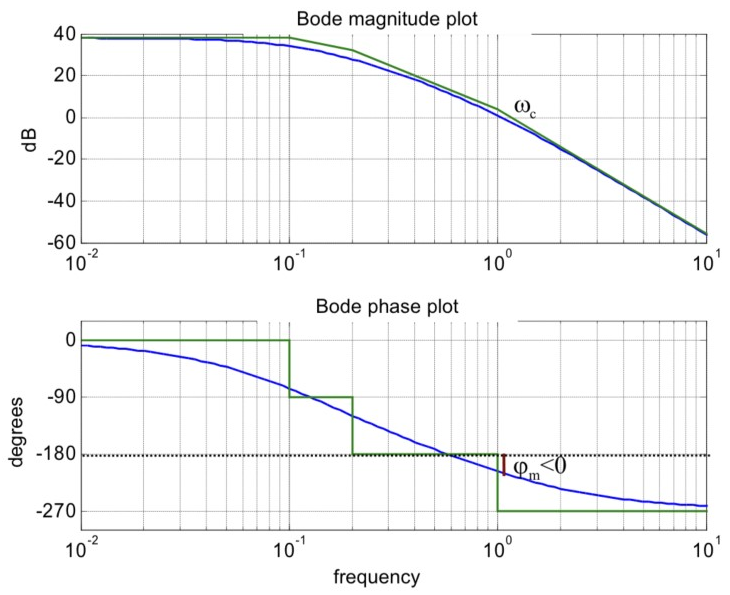
从本例的设计指标可知，我们要求|e(∞)|<=0.1，我们首先由误差部分开始控制器的设计。

由开环系统与闭环系统的近似关系可知，若控制器出现极点为零的情况，系统最终的误差e(∞)为零。而在第一种情况下，我们考虑控制器极点均在复平面的左半平面，即控制器的极点不包含零。我们假设控制器由两部分组成，即R(s)=R1(s)\*R2(s)，其中R1(s)为控制器的系数部分，R2(s)为控制器的零极点部分。

我们知道，e(∞)=ew(∞)+ed(∞)，其中，右侧等式的第一项是输入造成的误差，由输入信号确定，第二项是扰动造成的误差，由扰动信号确定。

例中已知：w(t)=A\*step(t), |A|<=1; dA(t)=B\*step(t), |B|<=5，从而可得：

e(∞)=ew(∞)+ed(∞)=(|A|+|B|)/(1+10\*R1(s))<=6/(1+10\*R1(s))<=0.1

从而求得：R1(s)>=5.9，我们选定R1(s)=8，接下来，我们先假设R2(s)为常数1，画出此时控制系统的波特图：

可以看到，该系统的相位裕量小于零，显然不满足我们的设计指标，因而我们需要通过调整R2(s)以满足指标。

第一种方案中，我们要先设法使得系统的截止频率与设计指标相同，且截止频率左侧有且仅有一个极点，我们已知系统在频率为零时幅值为38dB，这就要求：

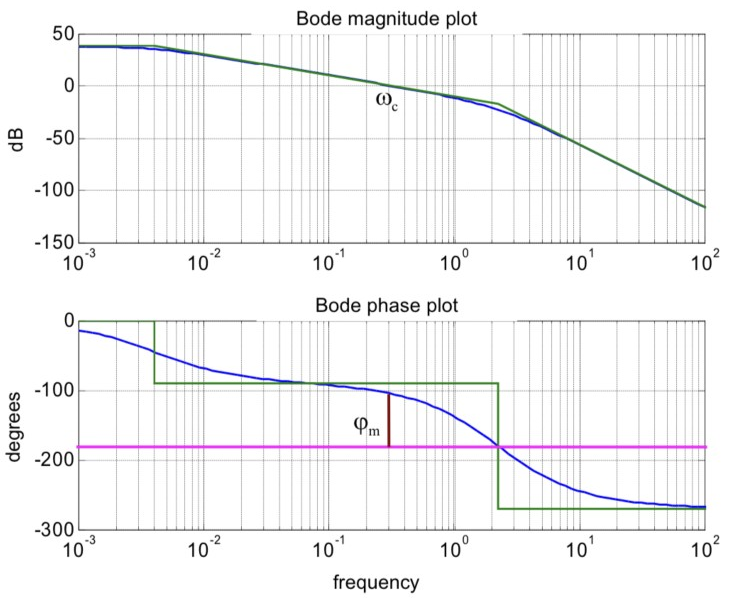
20\*lg(wc/wlow)=38dB=20\*lg80，即：wc/wlow=80，我们知道，wc为0.3，从而得到：wlow=0.004 rad/s。

为了保证修正后的波特图在频率变大时与原波特图相同，我们需要在高频率补偿零极点，从而使得两条曲线重合，这就要求在高频极点能够在两波特图交点处改变补偿后波特图的斜率，使得两线重合并一直延伸，或者说，R2(s)的幅值应为1，从而使得其并不会最终改变曲线特性，从而可得：

|R2(s)|=|(1+10s)(1+5s)(1+s)/((1+250s)(1+ts)^2)|=1，得到：t=0.5，从而得到case A的控制器:

R(s)=R1(s)\*R2(s)=8(1+10s)(1+5s)(1+s)/((1+250s)(1+0.5s)^2)

由此控制器与被控对象构成的控制系统波特图如下：



可以看到，第一种方案的控制器满足我们的基本设计指标，其基本频率为0.3，不是很大，但是相位裕量为77.1°，满足大于六十度的要求。

* 1. case B

第二种方案，我们仍考虑第一种方案中的题例。在第一种方案的基础上，我们改变R2(s)的选择策略，我们不再追求高频时修正过后的控制系统与原控制系统的波特图重合，而是将除了第一个极点外的其他极点尽可能消补到尽量高的频率去，从而保证系统的相位裕量足够大。本方案中，我们的控制器R(s)仍由两部分组成，系数项部分R1(s)的计算与第一种方案中相同，我们选取R1(s)=8。

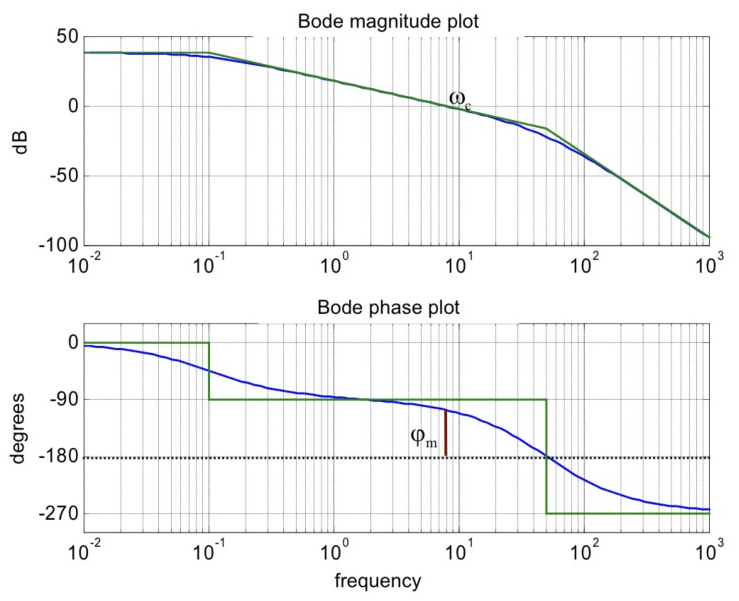
因为我们想消掉除第一个极点外的其他极点，并在高频中补偿，所以R2(s)应满足如下形式：

R2(s)=(1+5s)(1+s)/(1+ts)^2

其中t的取值由我们决定，这里我们选取补偿的高极点为50rad/s，可以得到：

R2(s)=(1+5s)(1+s)/(1+0.02s)^2，R(s)=R1(s)\*R2(s)=8(1+5s)(1+s)/(1+0.02s)^2

在此控制器下，控制器和被控对象组成的控制系统波特图如下：

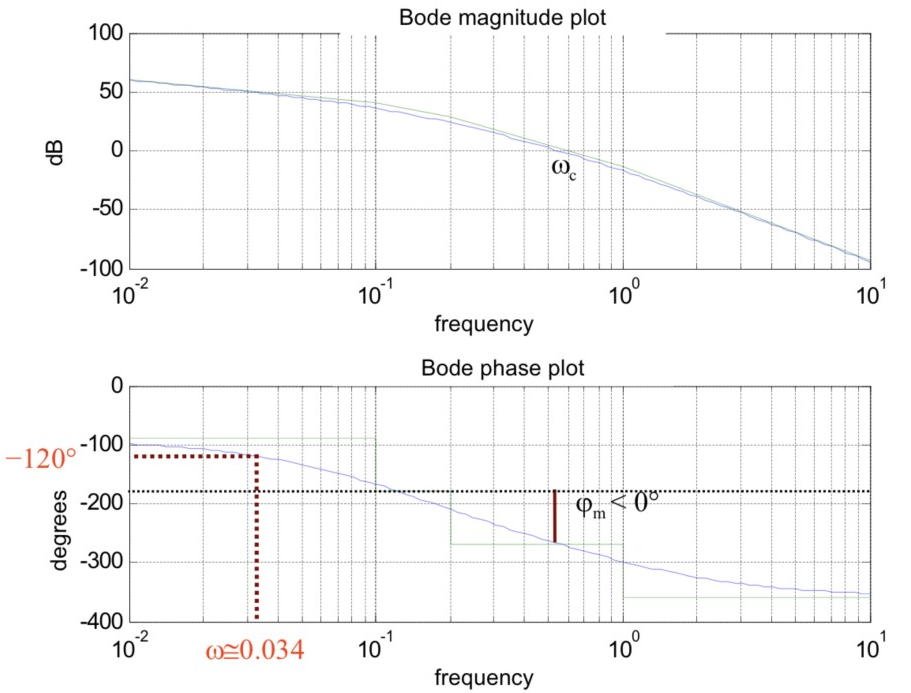


可以看到，第二种方案的控制器满足我们的基本设计指标，其基本频率为7.8，远大于要求所需，但是相位裕量为73°，相较于第一种方案，在截止频率大幅增加的情况下，相位裕量的改变不大，甚至略有下降。

* 1. case C

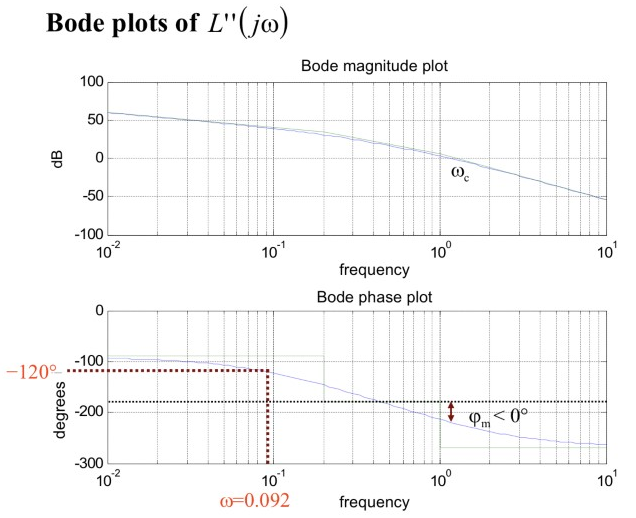
第三种方案，还是考虑原例，我们进一步增加对误差的要求，使得|e(∞)|=0，即R(s)=R1(s)\*R2(s)，且R1(s)=uR/s，R2(s)用以调整零极点，使得系统满足控制指标。

我们先令uR=R2(s)=1，画出该系统的波特图如下：



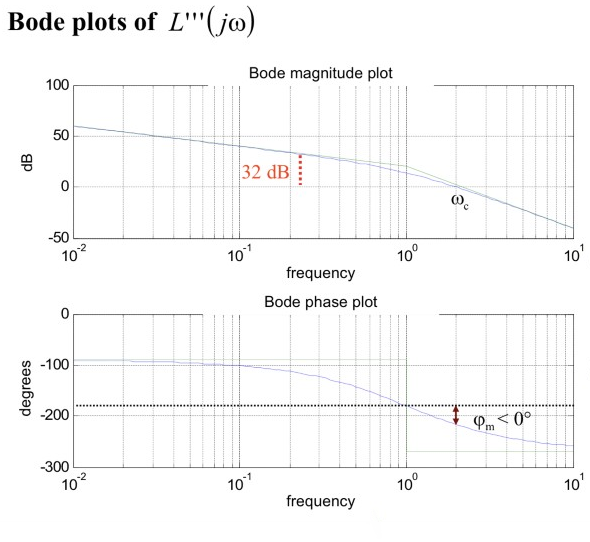
可以看到，系统的截止频率wc=0.55rad/s，但相位裕量仅有-88°，远远不满足要求，注意到当相位裕量刚好满足要求，即刚好等于60°时，w=0.034rad/s，当我们调整控制器的系数项以调整截止频率的值时，我们无法同时满足两个控制指标需求，因而我们需要调整控制器的R2(s)项，以使得两条件能同时满足。

我们首先消去被控对象的最小极点，即R2(s)=1+10s，注意到，因为我们的控制器是由两部分组成的，R(s)=R1(s)\*R2(s)，且R1(s)=uR/s，所以尽管R2(s)并未满足因果要求，但整个控制器是物理上可实现的，我们画出经过第一次调整后的系统波特图：

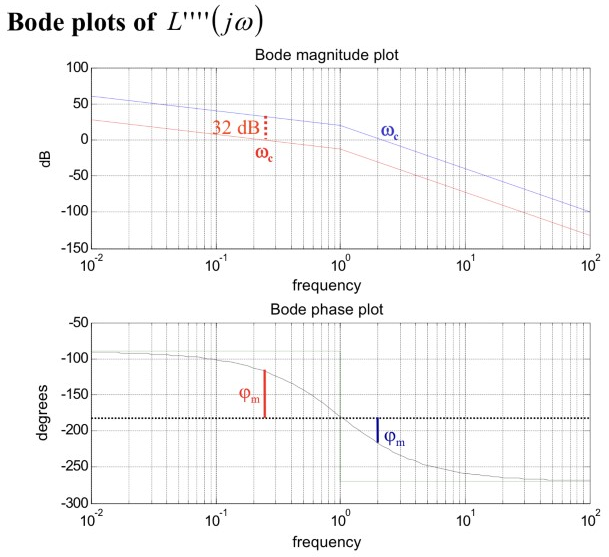


可以看到，系统的截止频率wc=1.15rad/s，但相位裕量仅有-39°，还是不满足要求，注意到当相位裕量刚好满足要求，即刚好等于60°时，w=0.092rad/s，当我们调整控制器的系数项以调整截止频率的值时，我们无法同时满足两个控制指标需求，因而我们还需要继续调整控制器的R2(s)项，以使得两条件能同时满足。

再一次尝试，我们再次减少一个被控对象的较小极点，并在较高频率进行补偿，此时R2(s)=(1+10s)(1+5s)/(1+s)，画出此时控制系统的波特图：



这时，系统的截止频率wc=2rad/s，但相位裕量仅有-37°，还是不满足要求，注意到当相位裕量刚好满足要求，即刚好等于60°时，w大于0.2rad/s，所以我们仅需调整控制器的系数项，以调整截止频率的值时，就可同时满足两个控制指标需求。当w=0.24rad/s时，|L(s)|=32dB，相位裕量也满足需求，所以我们需要使幅值的曲线下移，这时仅需uR=-32dB，即uR=10^(-32/20)=0.025，即可得到满足设计指标的控制系统，对比波特图如下：



此时控制器R(s)=R1(s)\*R2(s)=0.025(1+10s)(1+5s)/(s(1+s))。

第三种方案，我们得到了|e(∞)|=0的控制系统，但是相较于前两种方案，我们需要尝试的次数变多了，且计算量也增加了，同时因为零极点的缘故，我们的相位裕量仅为63°，相较前两种方案，相位裕量减小了。

可以看到，不同的方案选择计算量不同、控制器实现不同、系统性能指标不同，因而我们需要权衡利弊，在比较后选择最合适的方案。

1. 方案环境

在实现上，Python与MATLAB大同小异，但是考虑到Python的跨平台移植特性与其便捷性，本项目程序基于Python3.8版本，利用PyCharm平台实现，主要使用了control与sympy功能包。

Python3可由官方镜像直接下载安装，而PyCharm作为JetBrains众多工具软件中的一种，可用同济邮箱申请学生账户获取下载支持，其环境配置可根据需求选择与系统环境相同，或是创建虚拟环境，考虑到我们用到的Python模块很常用、便捷，我们选择将其安装配置在系统环境中，工程文件与系统环境一致。

control作为Python模块，可用于实现反馈控制系统分析和设计的基本操作，用以作为设计实现控制系统的模块，功能十分强大，它有如下功能特性：可实现状态空间和频域中的线性输入输出系统，可对系统的时间相应、频率响应做出特征分析，有完整的系统控制分析特征，包括其稳定性、可达性、可观测性、稳定裕度，还有线性滤波器等的控制接口。其中，我们主要用到了系统控制分析这一主要功能，包括如下几个函数：

ss(A, B, C, D[, dt])，以系数列表作为输入，用以创建状态空间系统；

tf(num, den[, dt])，用以创建传递函数系统，可以创建多输入多输出（MIMO）系统；

bode\_plot(syslist[, omega, Plot, …])，在（可选）频率范围内绘制系统的Bode图；

margin(sysdata)，用以计算增益和相位裕量以及相关频率；

pole(sys)，以传递函数或状态空间作为输入，输出系统的极点，用以分析系统特性；

zero(sys)，以传递函数或状态空间作为输入，输出系统的零点，用以分析系统特性；

sympy是一个科学计算库，可以利用一套强大的符号计算体系完成诸如多项式求值、求极限、解方程、求积分、微分方程、级数展开、矩阵运算等等计算问题，在这里我们主要使用sympy完成交互式系统的构建，使得我们能够输入自己想要完成的被控对象和控制指标，可以不再单单使用题例。我们主要使用了如下几个函数：

smpify (a，rational=False，evaluate=None)，用以简化表达式；

Symbol()，用以将字符转化成未知数；

pprint(expr)，用以将表达式标准化输出；

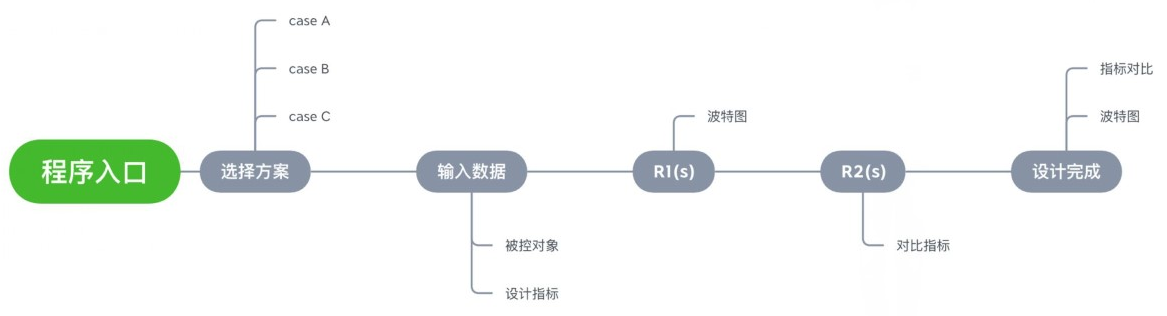
Poly(expr)，输入表达式，将其系数按列输出，用以得到表达式的系数因子；

control与sympy均可由pip3使用清华大学的国内镜像源直接下载安装，在linux或win系统下，我们可在终端中直接输入：pip install control/sympy进行环境的安装配置。

1. 设计实现

经过设计与编写，本次项目文件的代码架构如下：

├── src (项目源代码)  
│    ├── csddc.py (基类)  
│    ├── caseA.py (第一种方案)  
│    ├── caseB.py (第二种方案)  
│    └── caseC.py (第三种方案)  
│  
├── pics (项目运行图片)  
│    └── ```  
│  
├── bk (备份文件与开发文件)  
│    └── ```  
│  
├── main.py (项目入口)  
│  
└── readme.md (自述文件)

本项目为交互式设计，项目流程如图所示：

项目已上传到GitHub中，可访问链接查看：

1. 测试

This is a program for control system design simulation  
We have three different ways to do simulations, choose A, B, or C:

### case A

please input the gain part of the function:10

please input the zero part of the function:1

please input the pole part of the function:(1+s)\*(1+5\*s)\*(1+10\*s)

Your input function is:

10

────────────────

(s + 1)⋅(5⋅s + 1)⋅(10⋅s + 1)

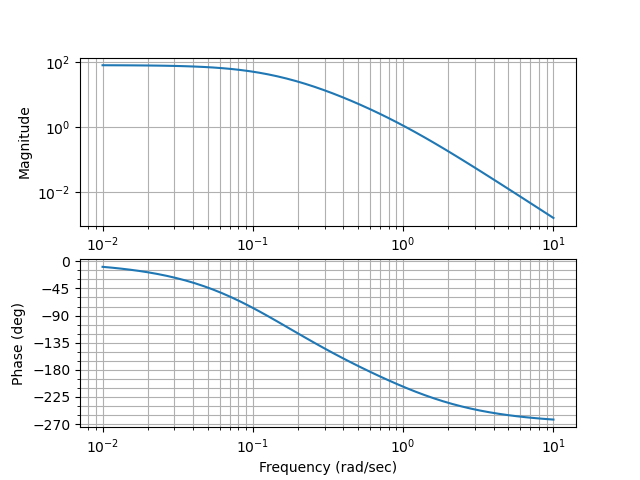
please input the phase limitation: 60

please input the frequency limitation: 0.2

Please input the amplitude of input step signal: 6

Please input the error rate when time goes to infinity: 0.1

In case A, R1 should be: 8



We can see from the first figure the condition is not satisfied

From the calculation we can get R2:

(s + 1)⋅(5⋅s + 1)⋅(10⋅s + 1)

─────────────────

2

(0.015625⋅s + 1) ⋅(400.0⋅s + 1)

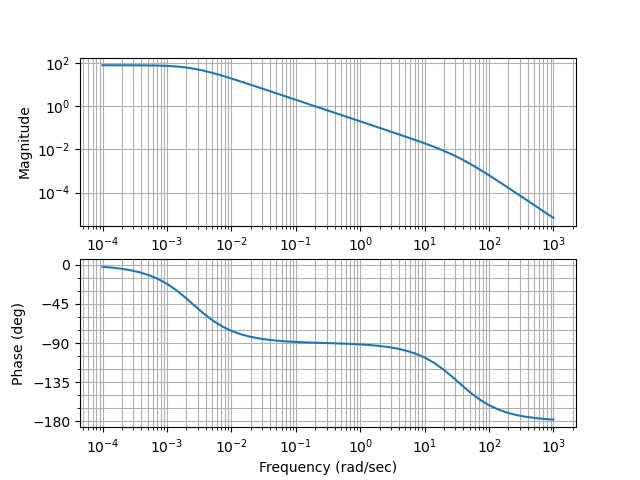
The ultimate function should be:

80

──────────────────

2

(0.015625⋅s + 1) ⋅(400.0⋅s + 1)



**After design simulation, the wp is: 0.19999577462578352**

**And the phase margin is: 90.3724397152805**

### case B

In case B, R1 should be: 8

And R2 should be:

(s + 1)⋅(5⋅s + 1)⋅(10⋅s + 1)

────────────────

2

(0.02⋅s + 1) ⋅(1.0⋅s + 1)

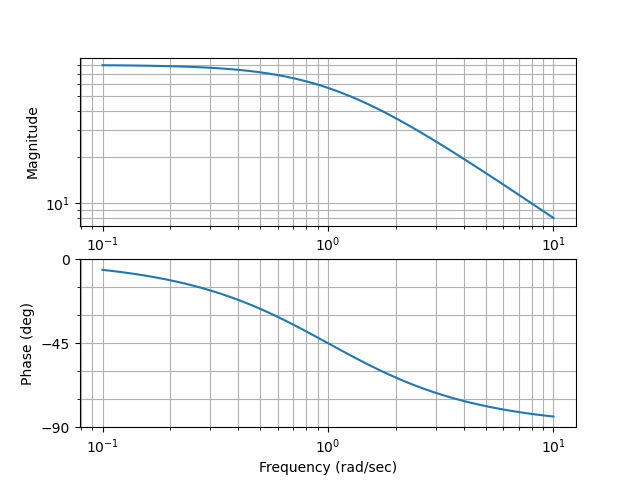
The ultimate function should be:

80

─────────────────────────

2

(0.02⋅s + 1) ⋅(1.0⋅s + 1)



**After design simulation, the wp is: 79.9937497558403**

**And the phase margin is: 90.71621589619497**

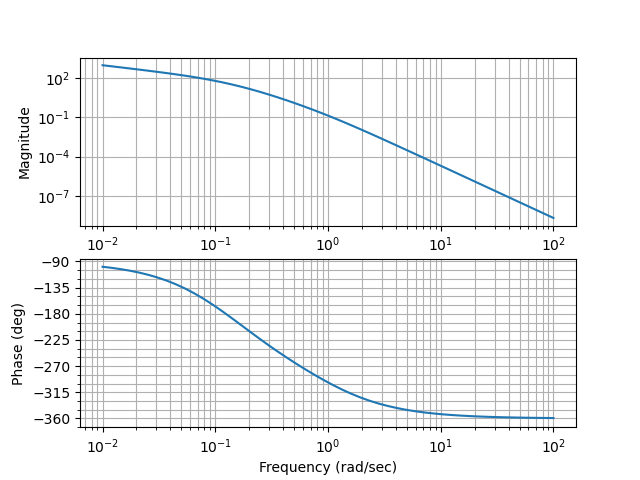
### Case C

R1 should be:

1

─

S

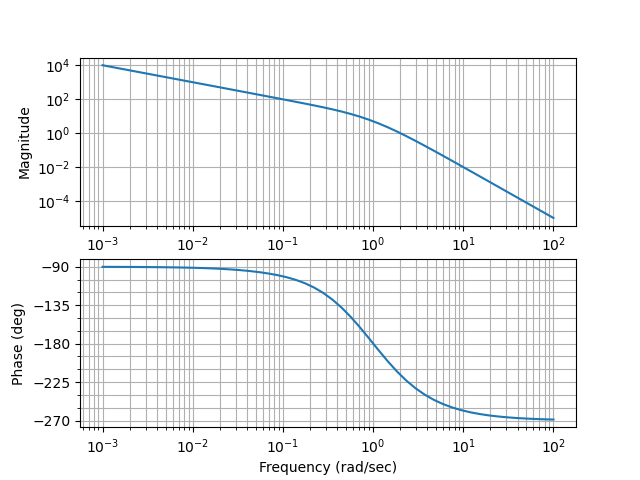


**phase margin at condition\_w: -29.744878733866187**

so it is impossible to just adjust the ur to satisfy the conditions

now we cancel the pole of G(s) in low frequency

R2 now is: 9.99999999999996\*s + 1



phase margin at condition\_w: 33.45903490936496

But the phase margin is not satisfied, we should adjust the wp

Now R2 is:

(5.00000000000001⋅s + 1)⋅(9.99999999999996⋅s + 1)

────────────────────────────

0.999999999999999⋅s + 1

when frequency equals to wc, the phase margin is: 68.02575257013994

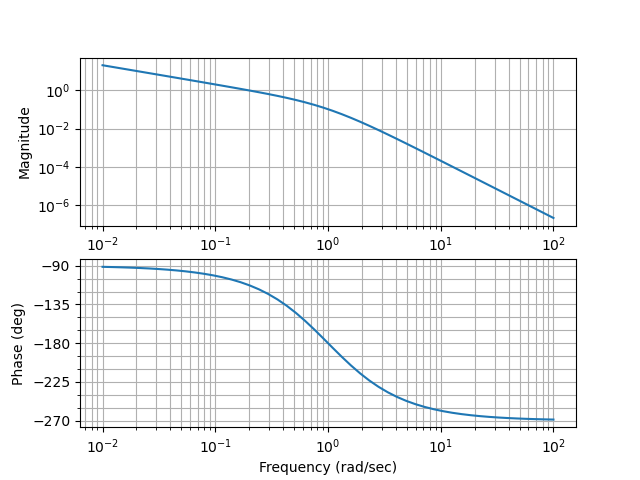
now the wp is: 2.0081903270569996 so we should adjust the wp to wc

The ultimate function should be:

10

─────────────────

(s + 1)⋅(5⋅s + 1)⋅(10⋅s + 1)



**After design simulation, the wp is: 0.19905100730450787**

**And the phase margin is: 67.48822866391856**

1. 思考与总结

本项目成功完成了三种不同方案下的最小相位系统设计实践，对零点极点为常规多项式的函数有较好的控制特性。本项目由如下几个优点：

1. 交互式设计，信息以会话方式输入，系统对用户的输入有直接反馈，处理的结果可以立刻被利用；
2. 实现了默认输入与自定义输入，能够实现的系统设计仿真不仅仅局限于课程内容中的题例，对其他输入也有较好处理；
3. 代码框架完整、注释清晰、可读性强，有较好的移植性，可在Linux和win上运行；
4. 自述文档完整、易于上手，代码依赖较少，对系统环境的要求较低；

但项目也存在一定的不足，因为python实现机制的原因，进行整数除法时结果并不精准，浮点数的一定量误差对系统函数有细微影响，但是因为其误差极小，对结果影响不大，主要体现在输出表达式的美观方面，例如分子上1.999996与分母的2无法相消。总的来说，本次课程项目为我们提供了认识并设计最小相位系统的实践机会，极大地帮助我们增进代码能力、思考能力、工程实践能力，我们学习到的知识可扩展性强，也为今后更多的实践打下了基础，是一次成功而优秀的课程考查。

1. 参考文献

Python Control Systems Library：https://python-control.readthedocs.io/en/0.8.3/

Github\_python\_control\_module：https://github.com/python-control/python-control

control · PyPI：https://pypi.org/project/control/0.7.0/

Python\_control\_example：https://coveralls.io/github/python-control/python-control

SymPy\_org：https://www.sympy.org/en/index.html

SymPy\_docs：https://docs.sympy.org/latest/index.html