Repressilator

Peter Gawthrop. *peter.gawthrop@unimelb.edu.au*

October 12, 2024

Contents

1 Introduction

This Jupyter notebook (Repressilator.ipynb) contains the code used to generate the following examples for the paper "Analysis of Biochemical Oscillators Using Bond Graphs and Linear Control Theory" by Peter Gawthrop and Michael Pan:

- 5. The Repressilator

The examples

- 3. Illustrative Example (system Toy)
- 4. The Sel'kov Oscillator (system Selkov)

are in the notebook Oscillation.ipynb

[1]:
```
import BondGraphTools as bgt
from BondGraphTools.reaction_builder import Reaction_Network
import numpy as np
import IPython.display as disp
import copy
import scipy.optimize as opt
import scipy.integrate as int

## Control systems package
import control as con
con.config.defaults['xferfcn.display_format'] = 'zpk'

## Stoichiometric analysis
import stoich as st

## BG from stoichiometry
import stoichBondGraph as stbg

# Allow output from within functions
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

#Plotting
import matplotlib.pyplot as plt
from cycler import cycler

## For reimporting: use imp.reload(module)
import importlib as imp

## Saving data
import pickle

quiet = True
Original = False
Plotting=False
SavingData = False
```

[2]:
```python
## System
SystemName = 'Repressilator'
```

[3]:
```python
## Reduced system order
redOrder = 3
largeOrder = 30
```

[4]:
```python
## Set up parameters to change
muA_nom_0 = 20
muA_nom = 1.0*muA_nom_0
```

[5]:
```python
def zero_crossings(a):
    """Zero crossings from  positive to negative
    """
    return np.where(np.diff(np.sign(a))>0)[0]
```

[6]:
```python
def rn2bg(rn,name):
    """
    Reaction network to Bond Graph.
    Creates BG file name.py
    Returns basic stoichiometry
    """
    # Extract stoichiometry from reaction network
    Nf_0 = np.array(rn.forward_stoichiometry)
    Nr_0 = np.array(rn.reverse_stoichiometry)
    N_0 = Nr_0 - Nf_0

    ## sanity check
    SanityCheck(N_0,rn.stoichiometry)

    ## Species and reactions
    species = rn.species
    n_X = len(species)
    reaction = []
    for Reaction in reactions:
        reaction.append(Reaction[1])

    ## Create BG from stoichiometry
    s_0 = {}
    s_0['N'] = N_0
    s_0['Nf'] = Nf_0
    s_0['Nr'] = Nr_0
    s_0['species'] = species
    s_0['n_X'] = len(species)
    s_0['reaction'] = reaction
    s_0['name'] = name
    stbg.model(s_0)

    return s_0
```

[7]:
```
# def SaveFig(SystemName,PlotName,Plotting=True,fontsize=14,linewidth=5):
#     if Plotting:
#         plt.rcParams.update({'font.size': fontsize})
#         plt.rcParams.update({'lines.linewidth': linewidth})
#         plotname = f'Figs/{SystemName}_{PlotName}.pdf'
#         plt.savefig(plotname)
```

[8]:
```
# def SaveFig(SystemName,PlotName,Plotting=True,fontsize=14,linewidth=5):
#     if Plotting:
#         plt.rcParams.update({'font.size': fontsize})
#         plt.rcParams.update({'lines.linewidth': linewidth})
#         plt.rcParams.update({'lines.markersize': 4*linewidth})
#         plotname = f'Figs/{SystemName}_{PlotName}.pdf'
#         plt.savefig(plotname)
```

[9]:
```
def SetPlot(fontsize=14,linewidth=5,RL=False):
        ## Sizes
        plt.rcParams.update({'font.size': fontsize})
        plt.rcParams.update({'lines.linewidth': linewidth})
        plt.rcParams.update({'lines.markersize': 6*linewidth})

        ## set up colour cycling for plot
        if RL:
            ## Root locus colors
            default_cycler = (cycler(color=['grey','grey','r', 'g', 'b']))
        else:
            default_cycler = (cycler(color=['r', 'g', 'b']))
        plt.rc('axes', prop_cycle=default_cycler)

SetPlot()
```

[10]:
```
def SaveFig(SystemName,PlotName,fontsize=14,linewidth=5,RL=False):
    if Plotting:
        SetPlot(RL=RL)
        plotname = f'Figs/{SystemName}_{PlotName}.pdf'
        plt.tight_layout()
        plt.savefig(plotname)
```

[11]:
```
def step_response(sys,T=None):
    resp = con.step_response(sys,T=T)
    t = resp.t
    y = np.array(resp.y).flatten()
    plt.plot(t,y)
    return y
```

[12]:
```
def impulse_response(sys,T=None):
    resp = con.impulse_response(sys,T=T)
    t = resp.t
    y = np.array(resp.y).flatten()
    plt.plot(t,y)
    return y
```

[13]:
```python
def initial_response(sys,T=None,x0=0):
    resp = con.initial_response(sys,T=T,x0=x0)
    t = resp.t
    y = np.array(resp.y).flatten()
    plt.plot(t,y)
    return y
```

[14]:
```python
def ExtractSubsystem(Sys,i_in,i_out):

    A_sub = Sys.A
    B_sub = Sys.B[:,i_in]
    C_sub = Sys.C[i_out,:]
    D_sub = Sys.D[i_out,i_in]

    return con.minreal(con.ss(A_sub,B_sub,C_sub,D_sub))
```

[15]:
```python
def SanityCheck(m_1,m_2):
    err = np.max(abs(np.array(m_1)-np.array(m_2)))
    if not err==0:
        print(f'Warning: sanity check failure. err={err}')
```

[16]:
```python
def extractSysdX(Sys,s,chemo,chemostats,outp):

    ## Index of output
    species = s['species']
    i_v = species.index(outp)

    ## Index of input
    i = chemostats.index(chemo)

    sys = con.ss(Sys.A,Sys.B[:,i],Sys.C[i_v,:],Sys.D[i_v,i])

    return sys

def␣
↪Lin(s,sc,parameter=None,x_ss=None,chemostats=[],outvar='dX',Inp=['P','E'],␣
↪Outp=['P','E'],quiet=True):

    ## Linearise
    SYS = st.lin(s,sc,x_ss=x_ss,parameter=parameter,outvar='dX',quiet=quiet)

    # Extract individual transfer functions
    TF = {}
    for inp in Inp:
        for outp in Outp:
            if not quiet:
                print(inp,'-->',outp)
            tf = con.tf(extractSysdX(SYS,s,inp,chemostats,outp))
            TF[f'{inp}_{outp}'] = tf
            if not quiet:
                print(tf)
```

```
    return TF
```

[17]:
```
def SteadyState(s,sc,parameter,x0,OutpVar='P3',Last=150,returnAll=False):

    t = np.linspace(0,Last,1000)
    ndat = st.sim(s,sc=sc,t=t,parameter=parameter,X0=x0,quiet=True)
    x_ss = ndat['X'][-1,:]
    # st.plot(s,ndat,dX=False,x_ss=x_ss,species=['P1','P2','P3'],reaction=[])
    # st.plot(s,ndat,dX=True,species=[OutpVar],reaction=[])

    ## Flow into P
    species = s['species']
    v_ss = ndat['dX'][-1,species.index(OutpVar)]
    print('v_ss=',v_ss)
    if returnAll:
        SS = {}
        ## Save up all steady-state data
        for key in ndat:
            if not key in ['t']:
#                   print(key)
                SS[key] = ndat[key][-1,:]
        return x_ss,v_ss,SS
    else:
        return x_ss,v_ss

def func(x_P):
    print('***', x_P)
    x0[species.index('P3')] = x_P
    x_ss,v_ss = SteadyState(s,OLsc,parameter,x0,OutpVar='P3')
    return v_ss

def findSteadyState(s,sc,parameter,x0,OutpVar='P3',returnAll=False):

    species = s['species']
    root = opt.fsolve(func,x0[species.index(OutpVar)],xtol=1e-3)

    x_P_ss = root[0]

    x0[species.index(OutpVar)] = x_P_ss

    if returnAll:
        x_ss,v_ss,SS =␣
↪SteadyState(s,sc,parameter,x0,OutpVar=OutpVar,returnAll=returnAll)
        return x_ss,SS
    else:
        x_ss,v_ss =␣
↪SteadyState(s,sc,parameter,x0,OutpVar=OutpVar,returnAll=returnAll)
        return x_ss,x_P_ss
```

```python
[18]: def IntegrateTF(L0,crite=1e-4):
          L0_tf = con.tf(L0)
          num0 = L0_tf.num[0][0]
          den0 = L0_tf.den[0][0]
      #     print(den0)
          ln = len(num0)
          if (abs(num0[ln-1])<crite):
              ## remove s factor in numerator
              num = num0[:ln-1]
              den = den0
          else:
              ## Integrator
              ld = len(den0)
              num = num0
              den = np.zeros(ld+1)
              den[:ld] = den0

          L = con.tf(num,den)
          return L
```

2 Define Model as Reaction Network and Create BondGraph-Tools Model

Define model

```python
[19]: reactions = [
          ("A + G1_E = G1_EA", "G1_Tc1"),
          ("G1_EA = G1_E + G1_M", "G1_Tc2"),
          ("G1_E + 2*P2 = G1_EI", "G1_Tc3"),
          ("G1_EA + 2*P2 = G1_EAI", "G1_Tc4"),
          ("G1_M + R = G1_C0", "G1_rb"),
          ("G1_C0 + A = G1_C1 + G1_M", "G1_r1"),
          ("G1_C1 + A = G1_C2", "G1_r2"),
          ("G1_C2 + A = G1_C3", "G1_r3"),
          ("G1_C3 + A = G1_C4", "G1_r4"),
          ("G1_C4 + A = G1_C5", "G1_r5"),
          ("G1_C5 + A = G1_C6", "G1_r6"),
          ("G1_C6 + A = G1_C7", "G1_r7"),
          ("G1_C7 + A = G1_C8", "G1_r8"),
          ("G1_C8 = R + P1", "G1_rt"),
          ("G1_M = G1_XM", "G1_degM"),
          ("P1 = G1_XP", "G1_degP"),

          ("A + G2_E = G2_EA", "G2_Tc1"),
          ("G2_EA = G2_E + G2_M", "G2_Tc2"),
          ("G2_E + 2*P3 = G2_EI", "G2_Tc3"),
          ("G2_EA + 2*P3 = G2_EAI", "G2_Tc4"),
          ("G2_M + R = G2_C0", "G2_rb"),
          ("G2_C0 + A = G2_C1 + G2_M", "G2_r1"),
          ("G2_C1 + A = G2_C2", "G2_r2"),
```

```
    ("G2_C2 + A = G2_C3", "G2_r3"),
    ("G2_C3 + A = G2_C4", "G2_r4"),
    ("G2_C4 + A = G2_C5", "G2_r5"),
    ("G2_C5 + A = G2_C6", "G2_r6"),
    ("G2_C6 + A = G2_C7", "G2_r7"),
    ("G2_C7 + A = G2_C8", "G2_r8"),
    ("G2_C8 = R + P2", "G2_rt"),
    ("G2_M = G2_XM", "G2_degM"),
    ("P2 = G2_XP", "G2_degP"),

    ("A + G3_E = G3_EA", "G3_Tc1"),
    ("G3_EA = G3_E + G3_M", "G3_Tc2"),
    ("G3_E + 2*P1 = G3_EI", "G3_Tc3"),
    ("G3_EA + 2*P1 = G3_EAI", "G3_Tc4"),
    ("G3_M + R = G3_C0", "G3_rb"),
    ("G3_C0 + A = G3_C1 + G3_M", "G3_r1"),
    ("G3_C1 + A = G3_C2", "G3_r2"),
    ("G3_C2 + A = G3_C3", "G3_r3"),
    ("G3_C3 + A = G3_C4", "G3_r4"),
    ("G3_C4 + A = G3_C5", "G3_r5"),
    ("G3_C5 + A = G3_C6", "G3_r6"),
    ("G3_C6 + A = G3_C7", "G3_r7"),
    ("G3_C7 + A = G3_C8", "G3_r8"),
    ("G3_C8 = R + P3", "G3_rt"),
    ("G3_M = G3_XM", "G3_degM"),
    ("P3 = G3_XP", "G3_degP"),
]

rn = Reaction_Network(name="RepressilatorFB")
for (reaction_string,reaction_name) in reactions:
    rn.add_reaction(reaction_string, name=reaction_name)

rn.add_chemostat("A")
rn.add_chemostat("G1_XM")
rn.add_chemostat("G1_XP")
rn.add_chemostat("G2_XM")
rn.add_chemostat("G2_XP")
rn.add_chemostat("G3_XM")
rn.add_chemostat("G3_XP")

if not Original:
    chemostats = ['A', 'G1_XM', 'G1_XP', 'G2_XM','G2_XP','G3_XM','G3_XP']
```

[20]:
```
if Original: # Use Michaels original version
    model = rn.as_network_model(normalised=True)
else:  # Use the stoichiometric aproach to build model
    s0 = rn2bg(rn,'RepressilatorFB_abg')
    species = s0['species']
    reaction = s0 ['reaction']
    n_X = s0['n_X']
```

```python
    import RepressilatorFB_abg as sys_abg
    model = sys_abg.model()
```

Set parameters

[21]:
```python
n = 8
alpha = np.exp(1.25)
mu_folding = 20.0
KP = alpha**n/np.exp(mu_folding)
mu0_P = n*np.log(alpha) - mu_folding
A_nom=5800000
# muA_nom=20
Kd = 30
h = 2

# Calculate thermodynamic constant from amount and chemical potential
def compute_K(x, mu):
    return (1 / x) * np.exp(mu)

# Thermodynamic constants
KR = 1.0
KM = 1.0
(model/"C:G1_M").set_param("k",KM)
(model/"C:G2_M").set_param("k",KM)
(model/"C:G3_M").set_param("k",KM)

# Transcription
w = 4.14
theta = 4.38
K_A = compute_K(A_nom, muA_nom)
r_Tc = w / (K_A * theta)
RbA = K_A * theta
RbM = 1e6
RbI = (100*KP)**h #Kd*mu0_P
r2 = (1+RbA/RbM)*r_Tc
r1 = r2*RbM/RbA
KE = 1
KEA = KE*RbA*r1/(r1+r2)
KEI = RbI*KE*KP
KEAI = RbI*KEA*KP
r3 = 1e6
r4 = 1e6
(rTc1, rTc2, rTc3, rTc4) = (r1, r2, r3, r4)

(model/"C:G1_E").set_param("k",KE)
(model/"C:G1_EA").set_param("k",KEA)
(model/"C:G1_EI").set_param("k",KEI)
(model/"C:G1_EAI").set_param("k",KEAI)
(model/"R:G1_Tc1").set_param("r",r1)
(model/"R:G1_Tc2").set_param("r",r2)
(model/"R:G1_Tc3").set_param("r",r3)
```

```
(model/"R:G1_Tc4").set_param("r",r4)

(model/"C:G2_E").set_param("k",KE)
(model/"C:G2_EA").set_param("k",KEA)
(model/"C:G2_EI").set_param("k",KEI)
(model/"C:G2_EAI").set_param("k",KEAI)
(model/"R:G2_Tc1").set_param("r",r1)
(model/"R:G2_Tc2").set_param("r",r2)
(model/"R:G2_Tc3").set_param("r",r3)
(model/"R:G2_Tc4").set_param("r",r4)

(model/"C:G3_E").set_param("k",KE)
(model/"C:G3_EA").set_param("k",KEA)
(model/"C:G3_EI").set_param("k",KEI)
(model/"C:G3_EAI").set_param("k",KEAI)
(model/"R:G3_Tc1").set_param("r",r1)
(model/"R:G3_Tc2").set_param("r",r2)
(model/"R:G3_Tc3").set_param("r",r3)
(model/"R:G3_Tc4").set_param("r",r4)

# Translation



γ_max = 1260
kf = 4 * γ_max  # Multiplied by 4 to account for 4 ATP molecules per amino␣
↪acid
KC0 = 1.0
(model/"C:G1_C0").set_param("k",KC0)
(model/"C:G2_C0").set_param("k",KC0)
(model/"C:G3_C0").set_param("k",KC0)
KC1 = alpha
(model/"C:G1_C1").set_param("k",KC1)
(model/"C:G2_C1").set_param("k",KC1)
(model/"C:G3_C1").set_param("k",KC1)
r1 = kf/np.exp(muA_nom)
(model/"R:G1_r1").set_param("r",r1)
(model/"R:G2_r1").set_param("r",r1)
(model/"R:G3_r1").set_param("r",r1)

for i in range(2, n+1):
    KCi = alpha**i
    (model/f"C:G1_C{i}").set_param("k",KCi)
    (model/f"C:G2_C{i}").set_param("k",KCi)
    (model/f"C:G3_C{i}").set_param("k",KCi)
    ri = kf/np.exp(muA_nom)/(alpha**(i-1))
    (model/f"R:G1_r{i}").set_param("r",ri)
    (model/f"R:G2_r{i}").set_param("r",ri)
    (model/f"R:G3_r{i}").set_param("r",ri)

rd = 100 * kf/(alpha**n)
```

```
(model/"R:G1_rt").set_param("r",rd)
(model/"R:G2_rt").set_param("r",rd)
(model/"R:G3_rt").set_param("r",rd)
rb = 1e-2
(model/"R:G1_rb").set_param("r",rb)
(model/"R:G2_rb").set_param("r",rb)
(model/"R:G3_rb").set_param("r",rb)

# Degradation
rdegM = np.log(2)/2
(model/"R:G1_degM").set_param("r",rdegM)
(model/"R:G2_degM").set_param("r",rdegM)
(model/"R:G3_degM").set_param("r",rdegM)

rdegP = np.log(2)/4/KP
(model/"R:G1_degP").set_param("r",rdegP)
(model/"R:G2_degP").set_param("r",rdegP)
(model/"R:G3_degP").set_param("r",rdegP)

(model/"C:R").set_param("k",1.0)
(model/"C:P1").set_param("k",KP)
(model/"C:P2").set_param("k",KP)
(model/"C:P3").set_param("k",KP)
```

[22]:

```
## Set chemostats
Large = 1e6
print(muA_nom)
mu_XM = np.log(1e-6)
mu_XP = np.log(1e-10)

if Original:
    (model/"SS:A").set_param("e",muA_nom)

    (model/"SS:G1_XM").set_param("e",mu_XM)
    (model/"SS:G2_XM").set_param("e",mu_XM)
    (model/"SS:G3_XM").set_param("e",mu_XM)

    (model/"SS:G1_XP").set_param("e",mu_XP)
    (model/"SS:G2_XP").set_param("e",mu_XP)
    (model/"SS:G3_XP").set_param("e",mu_XP)



else:
    (model/"C:A").set_param("k",np.exp(muA_nom)/Large)

    (model/"C:G1_XM").set_param("k",np.exp(mu_XM)/Large)
    (model/"C:G2_XM").set_param("k",np.exp(mu_XM)/Large)
    (model/"C:G3_XM").set_param("k",np.exp(mu_XM)/Large)

    (model/"C:G1_XP").set_param("k",np.exp(mu_XP)/Large)
```

```
    (model/"C:G2_XP").set_param("k",np.exp(mu_XP)/Large)
    (model/"C:G3_XP").set_param("k",np.exp(mu_XP)/Large)
```

```
20.0
```

3 Set parameters

[23]:
```
def SetParameter(Unit=False):
    parameter = {}
    for spec in species:
        comp = 'C:'+spec
        val = (model/comp).params['k']['value']
        if Unit:
            val=1
        if spec in chemostats:
    #           print('chemostat')
            # val = np.exp(val)
            print(f'Chemostat {spec}: {val:.2e}')
            if Unit:
                val=1
    #       print(f'{spec} \t{val:.2e}')
        name = f'K_{spec}'
        parameter[name] = val


    for reac in reaction:
        comp = 'R:'+reac
        val = (model/comp).params['r']['value']
        if Unit:
            val=1
    #       print(f'{reac} \t{val:.2e}')
        name = f'kappa_{reac}'
        parameter[name] = val

    return parameter

parameter = SetParameter()
```

```
Chemostat A: 4.85e+02
Chemostat G1_XM: 1.00e-12
Chemostat G1_XP: 1.00e-16
Chemostat G2_XM: 1.00e-12
Chemostat G2_XP: 1.00e-16
Chemostat G3_XM: 1.00e-12
Chemostat G3_XP: 1.00e-16
```

[24]:
```
## Write parameters to a function
f = open('SetParameterRepressilator.py','w')
chars = f.write('def SetParameterRepressilator():\n')
for key in parameter:
    str = f"parameter['{key}'] = {parameter[key]}\n"
```

```
#     print(str)
    chars = f.write(str)
f.close()
```

[25]:
```
# model.constitutive_relations
```

4 List of Parameters

[26]:
```
for par in parameter:
    print(f'{par} = {parameter[par]:.4g}')
```

```
K_A = 485.2
K_G1_E = 1
K_G1_EA = 366.2
K_G1_M = 1
K_P2 = 4.54e-05
K_G1_EI = 9.358e-10
K_G1_EAI = 3.427e-07
K_R = 1
K_G1_C0 = 1
K_G1_C1 = 3.49
K_G1_C2 = 12.18
K_G1_C3 = 42.52
K_G1_C4 = 148.4
K_G1_C5 = 518
K_G1_C6 = 1808
K_G1_C7 = 6311
K_G1_C8 = 2.203e+04
K_P1 = 4.54e-05
K_G1_XM = 1e-12
K_G1_XP = 1e-16
K_G2_E = 1
K_G2_EA = 366.2
K_G2_M = 1
K_P3 = 4.54e-05
K_G2_EI = 9.358e-10
K_G2_EAI = 3.427e-07
K_G2_C0 = 1
K_G2_C1 = 3.49
K_G2_C2 = 12.18
K_G2_C3 = 42.52
K_G2_C4 = 148.4
K_G2_C5 = 518
K_G2_C6 = 1808
K_G2_C7 = 6311
K_G2_C8 = 2.203e+04
K_G2_XM = 1e-12
K_G2_XP = 1e-16
K_G3_E = 1
K_G3_EA = 366.2
K_G3_M = 1
```

```
K_G3_EI = 9.358e-10
K_G3_EAI = 3.427e-07
K_G3_C0 = 1
K_G3_C1 = 3.49
K_G3_C2 = 12.18
K_G3_C3 = 42.52
K_G3_C4 = 148.4
K_G3_C5 = 518
K_G3_C6 = 1808
K_G3_C7 = 6311
K_G3_C8 = 2.203e+04
K_G3_XM = 1e-12
K_G3_XP = 1e-16
kappa_G1_Tc1 = 30.85
kappa_G1_Tc2 = 0.0113
kappa_G1_Tc3 = 1e+06
kappa_G1_Tc4 = 1e+06
kappa_G1_rb = 0.01
kappa_G1_r1 = 1.039e-05
kappa_G1_r2 = 2.976e-06
kappa_G1_r3 = 8.527e-07
kappa_G1_r4 = 2.443e-07
kappa_G1_r5 = 7e-08
kappa_G1_r6 = 2.005e-08
kappa_G1_r7 = 5.746e-09
kappa_G1_r8 = 1.646e-09
kappa_G1_rt = 22.88
kappa_G1_degM = 0.3466
kappa_G1_degP = 3817
kappa_G2_Tc1 = 30.85
kappa_G2_Tc2 = 0.0113
kappa_G2_Tc3 = 1e+06
kappa_G2_Tc4 = 1e+06
kappa_G2_rb = 0.01
kappa_G2_r1 = 1.039e-05
kappa_G2_r2 = 2.976e-06
kappa_G2_r3 = 8.527e-07
kappa_G2_r4 = 2.443e-07
kappa_G2_r5 = 7e-08
kappa_G2_r6 = 2.005e-08
kappa_G2_r7 = 5.746e-09
kappa_G2_r8 = 1.646e-09
kappa_G2_rt = 22.88
kappa_G2_degM = 0.3466
kappa_G2_degP = 3817
kappa_G3_Tc1 = 30.85
kappa_G3_Tc2 = 0.0113
kappa_G3_Tc3 = 1e+06
kappa_G3_Tc4 = 1e+06
kappa_G3_rb = 0.01
kappa_G3_r1 = 1.039e-05
```

```
kappa_G3_r2 = 2.976e-06
kappa_G3_r3 = 8.527e-07
kappa_G3_r4 = 2.443e-07
kappa_G3_r5 = 7e-08
kappa_G3_r6 = 2.005e-08
kappa_G3_r7 = 5.746e-09
kappa_G3_r8 = 1.646e-09
kappa_G3_rt = 22.88
kappa_G3_degM = 0.3466
kappa_G3_degP = 3817
```

5 Set initial conditions

[27]:
```
# model.state_vars
```

[28]:
```
def SetState(n_X):

    X0 = np.ones(n_X)
    small = 1e-6

    X0[species.index('G1_E')] =   0.25
    X0[species.index('G1_EA')] =   0.25
    X0[species.index('G1_M')] =   small
    X0[species.index('P2')] =   100.0
    X0[species.index('G1_EI')] =   0.25
    X0[species.index('G1_EAI')] =   0.25
    X0[species.index('R')] =   5000.0
    X0[species.index('G1_C0')] =   small
    X0[species.index('G1_C1')] =   small
    X0[species.index('G1_C2')] =   small
    X0[species.index('G1_C3')] =   small
    X0[species.index('G1_C4')] =   small
    X0[species.index('G1_C5')] =   small
    X0[species.index('G1_C6')] =   small
    X0[species.index('G1_C7')] =   small
    X0[species.index('G1_C8')] =   small
    X0[species.index('P1')] =   100.0
    X0[species.index('G2_E')] =   0.25
    X0[species.index('G2_EA')] =   0.25
    X0[species.index('G2_M')] =   small
    X0[species.index('P3')] =   1000.0
    X0[species.index('G2_EI')] =   0.25
    X0[species.index('G2_EAI')] =   0.25
    X0[species.index('G2_C0')] =   small
    X0[species.index('G2_C1')] =   small
    X0[species.index('G2_C2')] =   small
    X0[species.index('G2_C3')] =   small
    X0[species.index('G2_C4')] =   small
    X0[species.index('G2_C5')] =   small
    X0[species.index('G2_C6')] =   small
```

```
    X0[species.index('G2_C7')] =   small
    X0[species.index('G2_C8')] =   small
    X0[species.index('G3_E')] =   0.25
    X0[species.index('G3_EA')] =   0.25
    X0[species.index('G3_M')] =   small
    X0[species.index('G3_EI')] =   0.25
    X0[species.index('G3_EAI')] =   0.25
    X0[species.index('G3_C0')] =   small
    X0[species.index('G3_C1')] =   small
    X0[species.index('G3_C2')] =   small
    X0[species.index('G3_C3')] =   small
    X0[species.index('G3_C4')] =   small
    X0[species.index('G3_C5')] =   small
    X0[species.index('G3_C6')] =   small
    X0[species.index('G3_C7')] =   small
    X0[species.index('G3_C8')] =   small

    ## Reset chemostat states with large state so that ln Kx = mu
    for chem in chemostats:
        print('Resetting chemostat:', chem, ' to', Large)
        X0[species.index(chem)] = Large

    return X0
```

[29]:
```
if Original:
    x0 = np.array([
       0.25,   #C: G1_E
       0.25,   #C: G1_EA
       1e-6,   #C: G1_M
       100.0,   #C: P2
       0.25,   #C: G1_EI
       0.25,   #C: G1_EAI
       5000.0,   #C: R
       1e-6,   #C: G1_C0
       1e-6,   #C: G1_C1
       1e-6,   #C: G1_C2
       1e-6,   #C: G1_C3
       1e-6,   #C: G1_C4
       1e-6,   #C: G1_C5
       1e-6,   #C: G1_C6
       1e-6,   #C: G1_C7
       1e-6,   #C: G1_C8
       100.0,   #C: P1
       0.25,   #C: G2_E
       0.25,   #C: G2_EA,
       1e-6,   #C: G2_M,
       1000.0,   #C: P3
       0.25,   #C: G2_EI
       0.25,   #C: G2_EAI
       1e-6,   #C: G2_C0
       1e-6,   #C: G2_C1
```

```
        1e-6,  #C: G2_C2
        1e-6,  #C: G2_C3
        1e-6,  #C: G2_C4
        1e-6,  #C: G2_C5
        1e-6,  #C: G2_C6
        1e-6,  #C: G2_C7
        1e-6,  #C: G2_C8
        0.25,  #C: G3_E
        0.25,  #C: G3_EA
        1e-6,  #C: G3_M
        0.25,  #C: G3_EI
        0.25,  #C: G3_EAI
        1e-6,  #C: G3_C0
        1e-6,  #C: G3_C1
        1e-6,  #C: G3_C2
        1e-6,  #C: G3_C3
        1e-6,  #C: G3_C4
        1e-6,  #C: G3_C5
        1e-6,  #C: G3_C6
        1e-6,  #C: G3_C7
        1e-6,  #C: G3_C8
    ])
else:
    x0 = SetState(n_X)
```

```
Resetting chemostat: A  to 1000000.0
Resetting chemostat: G1_XM  to 1000000.0
Resetting chemostat: G1_XP  to 1000000.0
Resetting chemostat: G2_XM  to 1000000.0
Resetting chemostat: G2_XP  to 1000000.0
Resetting chemostat: G3_XM  to 1000000.0
Resetting chemostat: G3_XP  to 1000000.0
```

6 Simulate

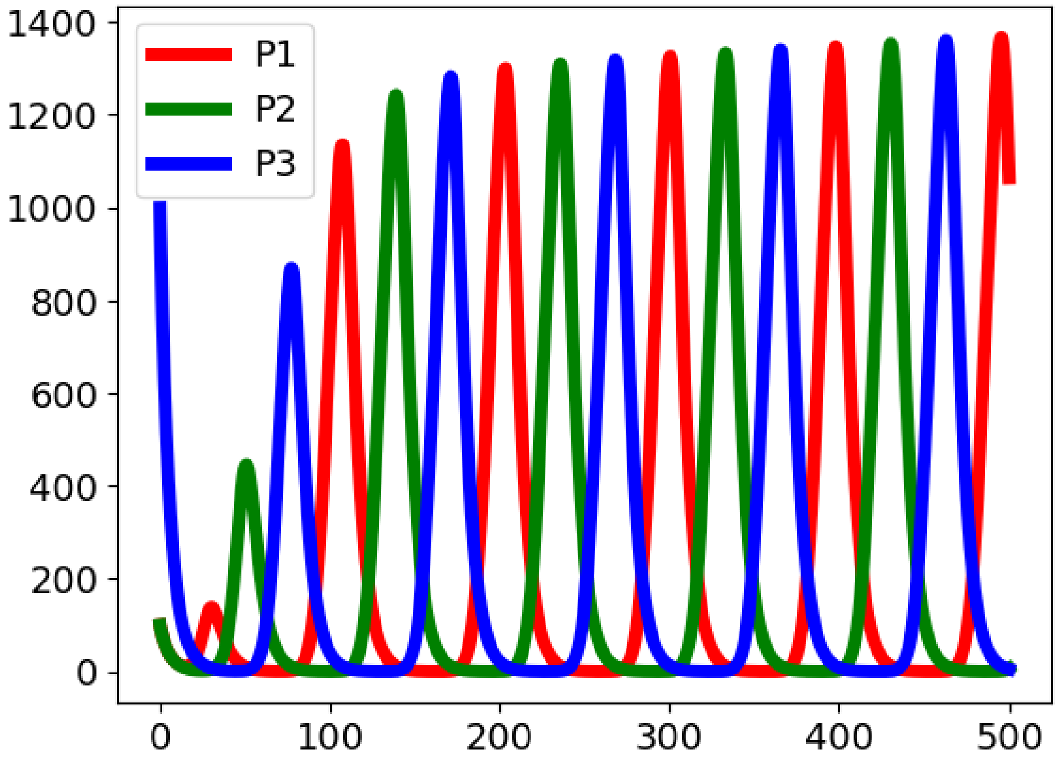
[30]:
```
timespan = [0.0, 500.0]
t, x = bgt.simulate(model, timespan=timespan, x0=x0, dt=1e-3)
```

[31]:
```
# Protein amounts
from matplotlib import pyplot as plt
plt.figure()
## Proteins
iP = [species.index('P1'), species.index('P2'), species.index('P3')]
## mRNA
iM = [species.index('G1_M'),species.index('G2_M'),species.index('G3_M')]
plt.plot(t,x[:,iP])
plt.legend(['P1','P2','P3'])
plt.show()
plt.plot(t,x[:,iM])
plt.legend(['M1','M2','M3'])
plt.show()
```

```
[31]: <Figure size 640x480 with 0 Axes>
```

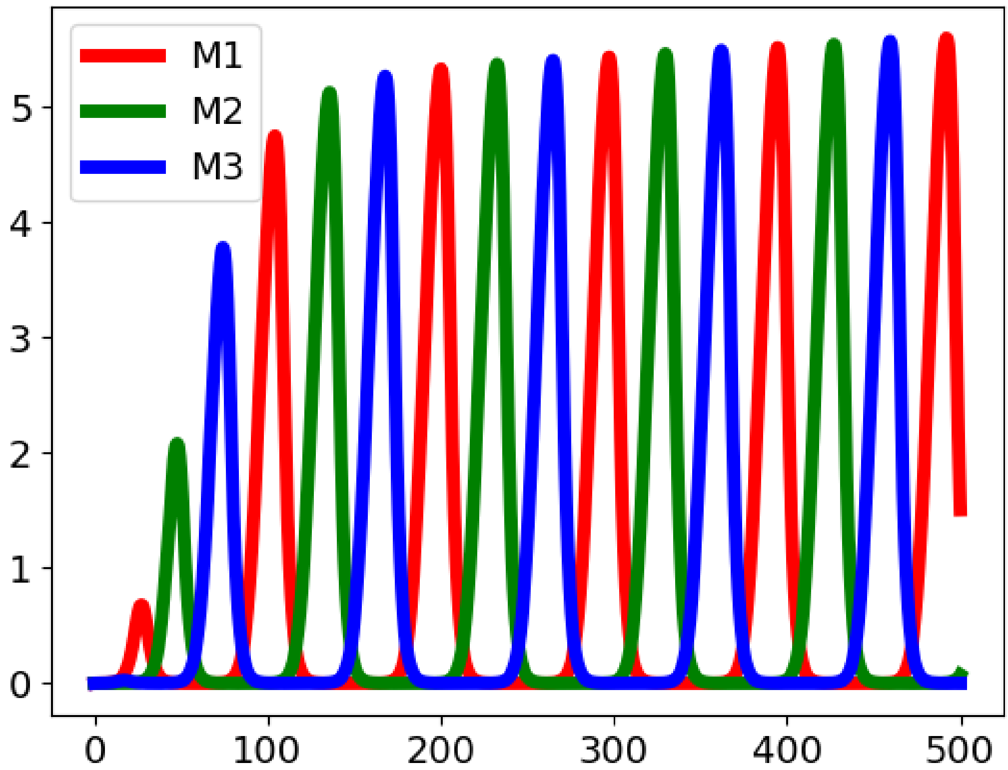
```
[31]: [<matplotlib.lines.Line2D at 0x7253a3e3ddc0>,
       <matplotlib.lines.Line2D at 0x7253a3e3de20>,
       <matplotlib.lines.Line2D at 0x7253a3e3de50>]
```

```
[31]: <matplotlib.legend.Legend at 0x7253a3dd7130>
```




```
[31]: [<matplotlib.lines.Line2D at 0x7253a3851640>,
       <matplotlib.lines.Line2D at 0x7253a38516a0>,
       <matplotlib.lines.Line2D at 0x7253a38435b0>]
```

```
[31]: <matplotlib.legend.Legend at 0x7253a3dd7160>
```



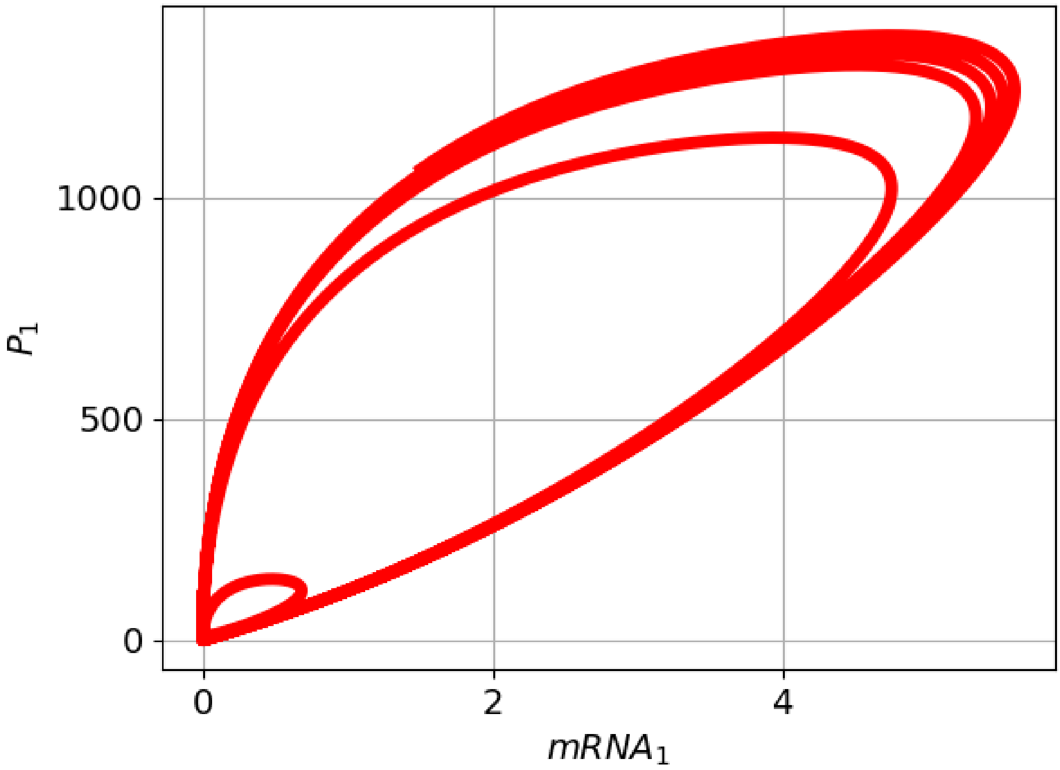
[32]:
```
# # mRNA amounts
# plt.figure()
# plt.plot(t,x[:,[2,19,34]])
# plt.show()
```

[33]:
```
plt.plot(x[:,iM[0]],x[:,iP[0]])
# plt.plot(x_ss[iM[0]],x_ss[iP[0]],marker='+')
plt.locator_params(nbins=4)
plt.grid()
plt.xlabel('$mRNA_1$')
plt.ylabel('$P_1$')
# SaveFig(SystemName,'PhasePlane')
```

[33]:
```
[<matplotlib.lines.Line2D at 0x7253a37c40d0>]
```

[33]:
```
Text(0.5, 0, '$mRNA_1$')
```

[33]:
```
Text(0, 0.5, '$P_1$')
```



7 Redo with stoichiometric approach

[34]:
```
if not Original:
## Recreate stoichiometry
    s = st.stoich(sys_abg.model(),quiet=quiet)
    N = s['N']
    Nf = s['Nf']
    Nr = s['Nr']

    ## Sanity check
    SanityCheck(N,s0['N'])
    SanityCheck(Nf,s0['Nf'])
    SanityCheck(Nr,s0['Nr'])
```

[35]:
```
## Chemostats
if not Original:
    sc = st.statify(s,chemostats=chemostats)
```

8 Save data - for use in Supplementary.ipynb

[36]:
```
if SavingData:
    ## Save data
    SavedData = {}
    StoichData = {}
```

```python
    StoichData['s'] = s
    StoichData['sc'] = sc
    StoichData['parameter'] = parameter
    SavedData['Stoich'] = StoichData

    SysName = 'Repressilator'
    file = open(f'{SysName}.dat', 'wb')
    pickle.dump(SavedData, file)
    file.close()
```

9 List of Reactions

[37]:
```python
## Reactions
disp.Latex(st.sprintrl(s,chemformula=False,split=10,all=True))
```

[37]:

$$A + G1E \Leftrightarrow G1EA \tag{1}$$
$$G1EA \Leftrightarrow G1E + G1M \tag{2}$$
$$G1E + 2P2 \Leftrightarrow G1EI \tag{3}$$
$$G1EA + 2P2 \Leftrightarrow G1EAI \tag{4}$$
$$G1M + R \Leftrightarrow G1C0 \tag{5}$$
$$A + G1C0 \Leftrightarrow G1M + G1C1 \tag{6}$$
$$A + G1C1 \Leftrightarrow G1C2 \tag{7}$$
$$A + G1C2 \Leftrightarrow G1C3 \tag{8}$$
$$A + G1C3 \Leftrightarrow G1C4 \tag{9}$$
$$A + G1C4 \Leftrightarrow G1C5 \tag{10}$$

$$A + G1C5 \Leftrightarrow G1C6 \tag{11}$$
$$A + G1C6 \Leftrightarrow G1C7 \tag{12}$$
$$A + G1C7 \Leftrightarrow G1C8 \tag{13}$$
$$G1C8 \Leftrightarrow R + P1 \tag{14}$$
$$G1M \Leftrightarrow G1XM \tag{15}$$
$$P1 \Leftrightarrow G1XP \tag{16}$$
$$A + G2E \Leftrightarrow G2EA \tag{17}$$
$$G2EA \Leftrightarrow G2E + G2M \tag{18}$$
$$G2E + 2P3 \Leftrightarrow G2EI \tag{19}$$
$$G2EA + 2P3 \Leftrightarrow G2EAI \tag{20}$$

$$R + G2M \Leftrightarrow G2C0 \tag{21}$$
$$A + G2C0 \Leftrightarrow G2M + G2C1 \tag{22}$$
$$A + G2C1 \Leftrightarrow G2C2 \tag{23}$$
$$A + G2C2 \Leftrightarrow G2C3 \tag{24}$$
$$A + G2C3 \Leftrightarrow G2C4 \tag{25}$$
$$A + G2C4 \Leftrightarrow G2C5 \tag{26}$$
$$A + G2C5 \Leftrightarrow G2C6 \tag{27}$$
$$A + G2C6 \Leftrightarrow G2C7 \tag{28}$$
$$A + G2C7 \Leftrightarrow G2C8 \tag{29}$$
$$G2C8 \Leftrightarrow P2 + R \tag{30}$$

$$G2M \Leftrightarrow G2XM \tag{31}$$
$$P2 \Leftrightarrow G2XP \tag{32}$$
$$A + G3E \Leftrightarrow G3EA \tag{33}$$
$$G3EA \Leftrightarrow G3E + G3M \tag{34}$$
$$2P1 + G3E \Leftrightarrow G3EI \tag{35}$$
$$2P1 + G3EA \Leftrightarrow G3EAI \tag{36}$$
$$R + G3M \Leftrightarrow G3C0 \tag{37}$$
$$A + G3C0 \Leftrightarrow G3M + G3C1 \tag{38}$$
$$A + G3C1 \Leftrightarrow G3C2 \tag{39}$$
$$A + G3C2 \Leftrightarrow G3C3 \tag{40}$$

$$A + G3C3 \Leftrightarrow G3C4 \tag{41}$$
$$A + G3C4 \Leftrightarrow G3C5 \tag{42}$$
$$A + G3C5 \Leftrightarrow G3C6 \tag{43}$$
$$A + G3C6 \Leftrightarrow G3C7 \tag{44}$$
$$A + G3C7 \Leftrightarrow G3C8 \tag{45}$$
$$G3C8 \Leftrightarrow R + P3 \tag{46}$$
$$G3M \Leftrightarrow G3XM \tag{47}$$
$$P3 \Leftrightarrow G3XP \tag{48}$$

10 List of Flows v as Function of States x

[38]:
```
## Flows
disp.Latex(st.sprintvl(s))
```

[38]:

$$v_{G1Tc1} = \kappa_{G1Tc1}\left(K_A K_{G1E} x_A x_{G1E} - K_{G1EA} x_{G1EA}\right) \tag{49}$$
$$v_{G1Tc2} = \kappa_{G1Tc2}\left(-K_{G1E} K_{G1M} x_{G1E} x_{G1M} + K_{G1EA} x_{G1EA}\right) \tag{50}$$
$$v_{G1Tc3} = \kappa_{G1Tc3}\left(K_{G1E} K_{P2}^2 x_{G1E} x_{P2}^2 - K_{G1EI} x_{G1EI}\right) \tag{51}$$
$$v_{G1Tc4} = \kappa_{G1Tc4}\left(K_{G1EA} K_{P2}^2 x_{G1EA} x_{P2}^2 - K_{G1EAI} x_{G1EAI}\right) \tag{52}$$
$$v_{G1rb} = \kappa_{G1rb}\left(-K_{G1C0} x_{G1C0} + K_{G1M} K_R x_{G1M} x_R\right) \tag{53}$$
$$v_{G1r1} = \kappa_{G1r1}\left(K_A K_{G1C0} x_A x_{G1C0} - K_{G1C1} K_{G1M} x_{G1C1} x_{G1M}\right) \tag{54}$$
$$v_{G1r2} = \kappa_{G1r2}\left(K_A K_{G1C1} x_A x_{G1C1} - K_{G1C2} x_{G1C2}\right) \tag{55}$$
$$v_{G1r3} = \kappa_{G1r3}\left(K_A K_{G1C2} x_A x_{G1C2} - K_{G1C3} x_{G1C3}\right) \tag{56}$$
$$v_{G1r4} = \kappa_{G1r4}\left(K_A K_{G1C3} x_A x_{G1C3} - K_{G1C4} x_{G1C4}\right) \tag{57}$$
$$v_{G1r5} = \kappa_{G1r5}\left(K_A K_{G1C4} x_A x_{G1C4} - K_{G1C5} x_{G1C5}\right) \tag{58}$$

$$v_{G1r6} = \kappa_{G1r6}\left(K_A K_{G1C5} x_A x_{G1C5} - K_{G1C6} x_{G1C6}\right) \tag{59}$$
$$v_{G1r7} = \kappa_{G1r7}\left(K_A K_{G1C6} x_A x_{G1C6} - K_{G1C7} x_{G1C7}\right) \tag{60}$$
$$v_{G1r8} = \kappa_{G1r8}\left(K_A K_{G1C7} x_A x_{G1C7} - K_{G1C8} x_{G1C8}\right) \tag{61}$$
$$v_{G1rt} = \kappa_{G1rt}\left(K_{G1C8} x_{G1C8} - K_{P1} K_R x_{P1} x_R\right) \tag{62}$$
$$v_{G1degM} = \kappa_{G1degM}\left(K_{G1M} x_{G1M} - K_{G1XM} x_{G1XM}\right) \tag{63}$$
$$v_{G1degP} = \kappa_{G1degP}\left(-K_{G1XP} x_{G1XP} + K_{P1} x_{P1}\right) \tag{64}$$
$$v_{G2Tc1} = \kappa_{G2Tc1}\left(K_A K_{G2E} x_A x_{G2E} - K_{G2EA} x_{G2EA}\right) \tag{65}$$
$$v_{G2Tc2} = \kappa_{G2Tc2}\left(-K_{G2E} K_{G2M} x_{G2E} x_{G2M} + K_{G2EA} x_{G2EA}\right) \tag{66}$$
$$v_{G2Tc3} = \kappa_{G2Tc3}\left(K_{G2E} K_{P3}^2 x_{G2E} x_{P3}^2 - K_{G2EI} x_{G2EI}\right) \tag{67}$$
$$v_{G2Tc4} = \kappa_{G2Tc4}\left(K_{G2EA} K_{P3}^2 x_{G2EA} x_{P3}^2 - K_{G2EAI} x_{G2EAI}\right) \tag{68}$$

$$v_{G2rb} = \kappa_{G2rb}\left(-K_{G2C0} x_{G2C0} + K_{G2M} K_R x_{G2M} x_R\right) \tag{69}$$
$$v_{G2r1} = \kappa_{G2r1}\left(K_A K_{G2C0} x_A x_{G2C0} - K_{G2C1} K_{G2M} x_{G2C1} x_{G2M}\right) \tag{70}$$
$$v_{G2r2} = \kappa_{G2r2}\left(K_A K_{G2C1} x_A x_{G2C1} - K_{G2C2} x_{G2C2}\right) \tag{71}$$
$$v_{G2r3} = \kappa_{G2r3}\left(K_A K_{G2C2} x_A x_{G2C2} - K_{G2C3} x_{G2C3}\right) \tag{72}$$
$$v_{G2r4} = \kappa_{G2r4}\left(K_A K_{G2C3} x_A x_{G2C3} - K_{G2C4} x_{G2C4}\right) \tag{73}$$
$$v_{G2r5} = \kappa_{G2r5}\left(K_A K_{G2C4} x_A x_{G2C4} - K_{G2C5} x_{G2C5}\right) \tag{74}$$
$$v_{G2r6} = \kappa_{G2r6}\left(K_A K_{G2C5} x_A x_{G2C5} - K_{G2C6} x_{G2C6}\right) \tag{75}$$
$$v_{G2r7} = \kappa_{G2r7}\left(K_A K_{G2C6} x_A x_{G2C6} - K_{G2C7} x_{G2C7}\right) \tag{76}$$
$$v_{G2r8} = \kappa_{G2r8}\left(K_A K_{G2C7} x_A x_{G2C7} - K_{G2C8} x_{G2C8}\right) \tag{77}$$
$$v_{G2rt} = \kappa_{G2rt}\left(K_{G2C8} x_{G2C8} - K_{P2} K_R x_{P2} x_R\right) \tag{78}$$

$$v_{G2degM} = \kappa_{G2degM}\left(K_{G2M}x_{G2M} - K_{G2XM}x_{G2XM}\right) \tag{79}$$
$$v_{G2degP} = \kappa_{G2degP}\left(-K_{G2XP}x_{G2XP} + K_{P2}x_{P2}\right) \tag{80}$$
$$v_{G3Tc1} = \kappa_{G3Tc1}\left(K_{A}K_{G3E}x_{A}x_{G3E} - K_{G3EA}x_{G3EA}\right) \tag{81}$$
$$v_{G3Tc2} = \kappa_{G3Tc2}\left(-K_{G3E}K_{G3M}x_{G3E}x_{G3M} + K_{G3EA}x_{G3EA}\right) \tag{82}$$
$$v_{G3Tc3} = \kappa_{G3Tc3}\left(K_{G3E}K_{P1}^{2}x_{G3E}x_{P1}^{2} - K_{G3EI}x_{G3EI}\right) \tag{83}$$
$$v_{G3Tc4} = \kappa_{G3Tc4}\left(K_{G3EA}K_{P1}^{2}x_{G3EA}x_{P1}^{2} - K_{G3EAI}x_{G3EAI}\right) \tag{84}$$
$$v_{G3rb} = \kappa_{G3rb}\left(-K_{G3C0}x_{G3C0} + K_{G3M}K_{R}x_{G3M}x_{R}\right) \tag{85}$$
$$v_{G3r1} = \kappa_{G3r1}\left(K_{A}K_{G3C0}x_{A}x_{G3C0} - K_{G3C1}K_{G3M}x_{G3C1}x_{G3M}\right) \tag{86}$$
$$v_{G3r2} = \kappa_{G3r2}\left(K_{A}K_{G3C1}x_{A}x_{G3C1} - K_{G3C2}x_{G3C2}\right) \tag{87}$$
$$v_{G3r3} = \kappa_{G3r3}\left(K_{A}K_{G3C2}x_{A}x_{G3C2} - K_{G3C3}x_{G3C3}\right) \tag{88}$$

$$v_{G3r4} = \kappa_{G3r4}\left(K_{A}K_{G3C3}x_{A}x_{G3C3} - K_{G3C4}x_{G3C4}\right) \tag{89}$$
$$v_{G3r5} = \kappa_{G3r5}\left(K_{A}K_{G3C4}x_{A}x_{G3C4} - K_{G3C5}x_{G3C5}\right) \tag{90}$$
$$v_{G3r6} = \kappa_{G3r6}\left(K_{A}K_{G3C5}x_{A}x_{G3C5} - K_{G3C6}x_{G3C6}\right) \tag{91}$$
$$v_{G3r7} = \kappa_{G3r7}\left(K_{A}K_{G3C6}x_{A}x_{G3C6} - K_{G3C7}x_{G3C7}\right) \tag{92}$$
$$v_{G3r8} = \kappa_{G3r8}\left(K_{A}K_{G3C7}x_{A}x_{G3C7} - K_{G3C8}x_{G3C8}\right) \tag{93}$$
$$v_{G3rt} = \kappa_{G3rt}\left(K_{G3C8}x_{G3C8} - K_{P3}K_{R}x_{P3}x_{R}\right) \tag{94}$$
$$v_{G3degM} = \kappa_{G3degM}\left(K_{G3M}x_{G3M} - K_{G3XM}x_{G3XM}\right) \tag{95}$$
$$v_{G3degP} = \kappa_{G3degP}\left(-K_{G3XP}x_{G3XP} + K_{P3}x_{P3}\right) \tag{96}$$

11 List of State x Derivatives as Function of Flows v.

[39]:
```
## State equations
imp.reload(st)
eqns = st.sprintdxl(s,sc)
disp.Latex(eqns)
```

[39]: <module 'stoich' from '/home/peterg/WORK/Research/SystemsBiology/lib/python/stoich.py'>

[39]:

$$\dot{x}_{G1E} = -v_{G1Tc1} + v_{G1Tc2} - v_{G1Tc3} \tag{97}$$
$$\dot{x}_{G1EA} = v_{G1Tc1} - v_{G1Tc2} - v_{G1Tc4} \tag{98}$$
$$\dot{x}_{G1M} = v_{G1Tc2} - v_{G1rb} + v_{G1r1} - v_{G1degM} \tag{99}$$
$$\dot{x}_{P2} = -2v_{G1Tc3} - 2v_{G1Tc4} + v_{G2rt} - v_{G2degP} \tag{100}$$
$$\dot{x}_{G1EI} = v_{G1Tc3} \tag{101}$$
$$\dot{x}_{G1EAI} = v_{G1Tc4} \tag{102}$$
$$\dot{x}_{R} = -v_{G1rb} + v_{G1rt} - v_{G2rb} + v_{G2rt} - v_{G3rb} + v_{G3rt} \tag{103}$$
$$\dot{x}_{G1C0} = v_{G1rb} - v_{G1r1} \tag{104}$$
$$\dot{x}_{G1C1} = v_{G1r1} - v_{G1r2} \tag{105}$$
$$\dot{x}_{G1C2} = v_{G1r2} - v_{G1r3} \tag{106}$$

$$\dot{x}_{G1C3} = v_{G1r3} - v_{G1r4} \tag{107}$$
$$\dot{x}_{G1C4} = v_{G1r4} - v_{G1r5} \tag{108}$$
$$\dot{x}_{G1C5} = v_{G1r5} - v_{G1r6} \tag{109}$$
$$\dot{x}_{G1C6} = v_{G1r6} - v_{G1r7} \tag{110}$$
$$\dot{x}_{G1C7} = v_{G1r7} - v_{G1r8} \tag{111}$$
$$\dot{x}_{G1C8} = v_{G1r8} - v_{G1rt} \tag{112}$$
$$\dot{x}_{P1} = v_{G1rt} - v_{G1degP} - 2v_{G3Tc3} - 2v_{G3Tc4} \tag{113}$$
$$\dot{x}_{G2E} = -v_{G2Tc1} + v_{G2Tc2} - v_{G2Tc3} \tag{114}$$
$$\dot{x}_{G2EA} = v_{G2Tc1} - v_{G2Tc2} - v_{G2Tc4} \tag{115}$$
$$\dot{x}_{G2M} = v_{G2Tc2} - v_{G2rb} + v_{G2r1} - v_{G2degM} \tag{116}$$

$$\dot{x}_{P3} = -2v_{G2Tc3} - 2v_{G2Tc4} + v_{G3rt} - v_{G3degP} \tag{117}$$
$$\dot{x}_{G2EI} = v_{G2Tc3} \tag{118}$$
$$\dot{x}_{G2EAI} = v_{G2Tc4} \tag{119}$$
$$\dot{x}_{G2C0} = v_{G2rb} - v_{G2r1} \tag{120}$$
$$\dot{x}_{G2C1} = v_{G2r1} - v_{G2r2} \tag{121}$$
$$\dot{x}_{G2C2} = v_{G2r2} - v_{G2r3} \tag{122}$$
$$\dot{x}_{G2C3} = v_{G2r3} - v_{G2r4} \tag{123}$$
$$\dot{x}_{G2C4} = v_{G2r4} - v_{G2r5} \tag{124}$$
$$\dot{x}_{G2C5} = v_{G2r5} - v_{G2r6} \tag{125}$$
$$\dot{x}_{G2C6} = v_{G2r6} - v_{G2r7} \tag{126}$$

$$\dot{x}_{G2C7} = v_{G2r7} - v_{G2r8} \tag{127}$$
$$\dot{x}_{G2C8} = v_{G2r8} - v_{G2rt} \tag{128}$$
$$\dot{x}_{G3E} = -v_{G3Tc1} + v_{G3Tc2} - v_{G3Tc3} \tag{129}$$
$$\dot{x}_{G3EA} = v_{G3Tc1} - v_{G3Tc2} - v_{G3Tc4} \tag{130}$$
$$\dot{x}_{G3M} = v_{G3Tc2} - v_{G3rb} + v_{G3r1} - v_{G3degM} \tag{131}$$
$$\dot{x}_{G3EI} = v_{G3Tc3} \tag{132}$$
$$\dot{x}_{G3EAI} = v_{G3Tc4} \tag{133}$$
$$\dot{x}_{G3C0} = v_{G3rb} - v_{G3r1} \tag{134}$$
$$\dot{x}_{G3C1} = v_{G3r1} - v_{G3r2} \tag{135}$$
$$\dot{x}_{G3C2} = v_{G3r2} - v_{G3r3} \tag{136}$$

$$\dot{x}_{G3C3} = v_{G3r3} - v_{G3r4} \tag{137}$$
$$\dot{x}_{G3C4} = v_{G3r4} - v_{G3r5} \tag{138}$$
$$\dot{x}_{G3C5} = v_{G3r5} - v_{G3r6} \tag{139}$$
$$\dot{x}_{G3C6} = v_{G3r6} - v_{G3r7} \tag{140}$$
$$\dot{x}_{G3C7} = v_{G3r7} - v_{G3r8} \tag{141}$$
$$\dot{x}_{G3C8} = v_{G3r8} - v_{G3rt} \tag{142}$$

12 Simulate (using stoichiometric approach)

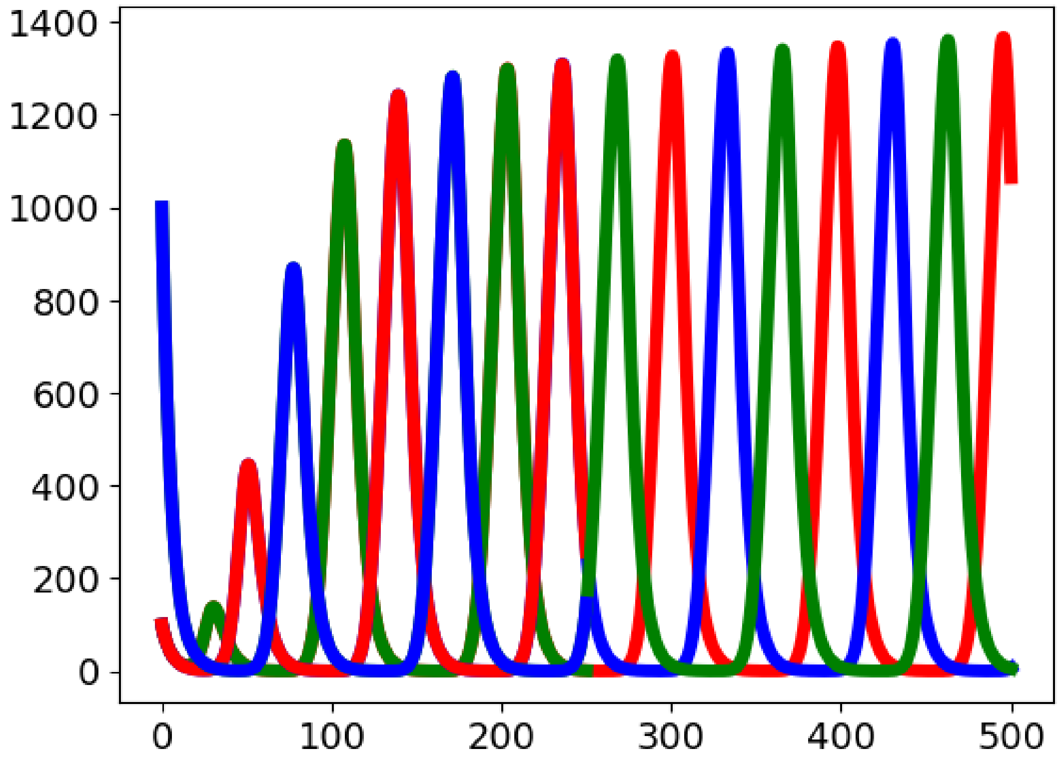
[40]:
```python
if not Original:


    ## Simulate
    tt = np.linspace(0,250,10000)
    ndat = st.
 ↪sim(s,sc,reduced=False,t=tt,X0=x0,parameter=parameter,quiet=True)
```

[41]:
```python
## Sizes
species = s['species']
print(f'Number of species = {len(species)} of which {len(chemostats)} are␣
 ↪chemostats giving {len(species)-len(chemostats)} states.')
```

```
Number of species = 53 of which 7 are chemostats giving 46 states.
```

[42]:
```python
# Protein amounts
iP = [species.index('P1'), species.index('P2'), species.index('P3')]
for i in iP:
    plt.plot(t,x[:,i])
    plt.plot(tt,ndat['X'][:,i])

plt.show()
```

```
[42]: [<matplotlib.lines.Line2D at 0x7253de9ba460>]

[42]: [<matplotlib.lines.Line2D at 0x7253f0309b20>]

[42]: [<matplotlib.lines.Line2D at 0x7253a3f3ee80>]

[42]: [<matplotlib.lines.Line2D at 0x7253de9ba220>]

[42]: [<matplotlib.lines.Line2D at 0x7253a3c477f0>]

[42]: [<matplotlib.lines.Line2D at 0x7253a3767ee0>]
```



[43]:
```
# st.plot(s,ndat,species = ['P1'],reaction = [])
# st.plot(s,ndat,species = ['P2'],reaction = [])
# st.plot(s,ndat,species = ['P3'],reaction = [])
# print(species.index('P3'))
```

13 Power

[44]:
```
P_Re = ndat['P_Re']
P_C = -ndat['P_C']
i_chemo = []
for chemo in chemostats:
    i_chemo.append(species.index(chemo))

free = list(set(species)-set(chemostats))
print(free)
i_free = []
for fr in free:
    i_free.append(species.index(fr))

P_chemo = P_C[:,i_chemo]
P_free = P_C[:,i_free]

## Total power
PP_Re = np.sum(P_Re,axis=1)
PP_C = np.sum(P_C,axis=1)
```

```
PP_chemo = np.sum(P_chemo,axis=1)
PP_free = np.sum(P_free,axis=1)
```

```
['G3_C2', 'G3_C7', 'G1_E', 'G1_EA', 'G3_C4', 'G1_C2', 'G2_EI', 'G1_EAI', 'P1',
'G1_C0', 'G2_C6', 'G3_C5', 'G3_C8', 'G2_E', 'G1_C5', 'G2_C2', 'G3_C3', 'G2_EA',
'G1_EI', 'G1_C7', 'G1_C1', 'G2_C1', 'G3_E', 'G3_EAI', 'G3_EI', 'G1_M', 'G1_C3',
'P2', 'G3_EA', 'G1_C6', 'R', 'G2_C5', 'G2_C4', 'G3_C0', 'G1_C4', 'G2_C8', 'P3',
'G2_C3', 'G3_C1', 'G3_C6', 'G2_EAI', 'G2_C0', 'G3_M', 'G1_C8', 'G2_M', 'G2_C7']
```

[45]:
```
t = ndat['t']
print(t)
t0 = 0.1
i0, = np.where(np.isclose(t, t0,atol=(t[1]-t[0])/2))[0]
print(i0,t[i0])
print(t[i0:])
plt.plot(t[i0:],PP_Re[i0:],label='Re',lw=10)
plt.plot(t[i0:],PP_free[i0:],label='Ce (free)',lw=5)
plt.plot(t[i0:],PP_chemo[i0:],label='chemostats',lw=2)
plt.legend()
plt.xlabel('$t$')
plt.ylabel('$p/kT$')

SaveFig(SystemName,'Power')
```

```
[0.00000000e+00 2.50025003e-02 5.00050005e-02 ... 2.49949995e+02
 2.49974997e+02 2.50000000e+02]
4 0.1000100010001
[1.00010001e-01 1.25012501e-01 1.50015002e-01 ... 2.49949995e+02
 2.49974997e+02 2.50000000e+02]
```

[45]: `[<matplotlib.lines.Line2D at 0x7253c420be50>]`

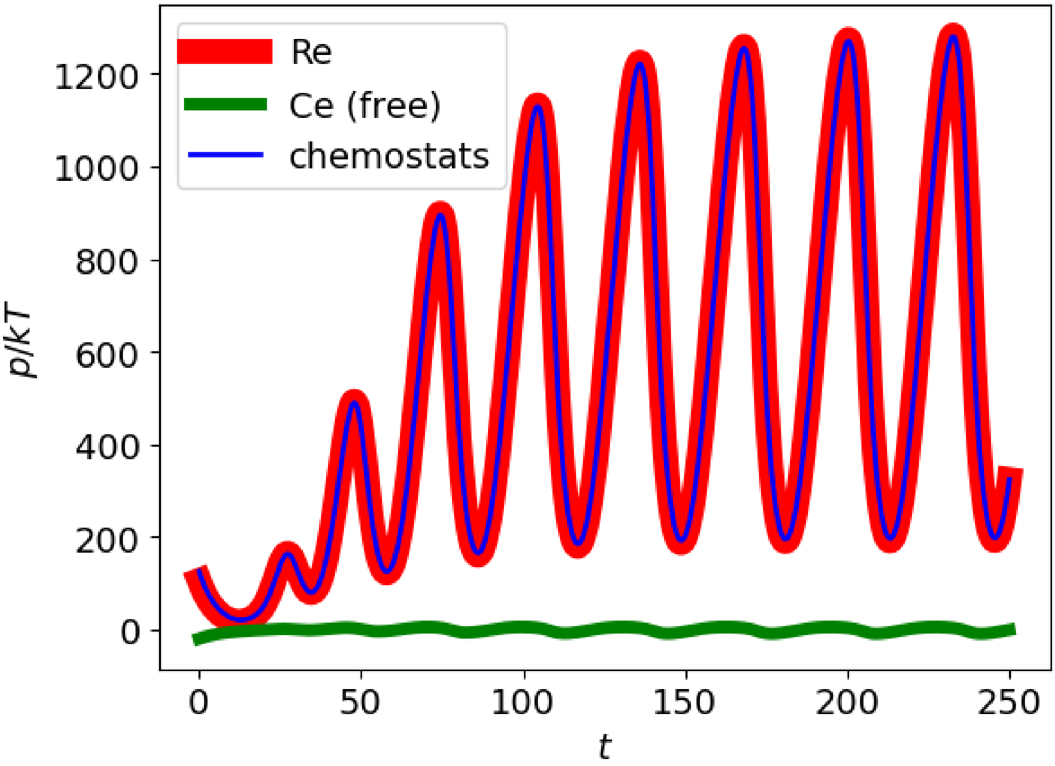
[45]: `[<matplotlib.lines.Line2D at 0x7253c420bc10>]`

[45]: `[<matplotlib.lines.Line2D at 0x7253c4844820>]`

[45]: `<matplotlib.legend.Legend at 0x7253de954190>`

[45]: `Text(0.5, 0, '$t$')`

[45]: `Text(0, 0.5, '$p/kT$')`



[46]:
```
t0 = 100
i0, = np.where(np.isclose(t, t0,atol=(t[1]-t[0])/2))[0]
print(i0,t[i0])
```

```
4000 100.0100010001
```

[47]:
```
imp.reload(st)
kT =st.kT()
print(f'{kT*1e21:.4} zJ')
```

[47]:
```
<module 'stoich' from
'/home/peterg/WORK/Research/SystemsBiology/lib/python/stoich.py'>
```

```
4.282 zJ
```

14 Steady state

[48]:
```
##Fix P3
OLchemostats = copy.copy(chemostats)
OLchemostats.append('P3')
print(OLchemostats)
OLsc = st.statify(s,chemostats=OLchemostats)

## Guess a vaue for x_P3 to give zero flow
guess = 10.75855
# Guess = guess+0.0001*np.linspace(-1,1,5)
```

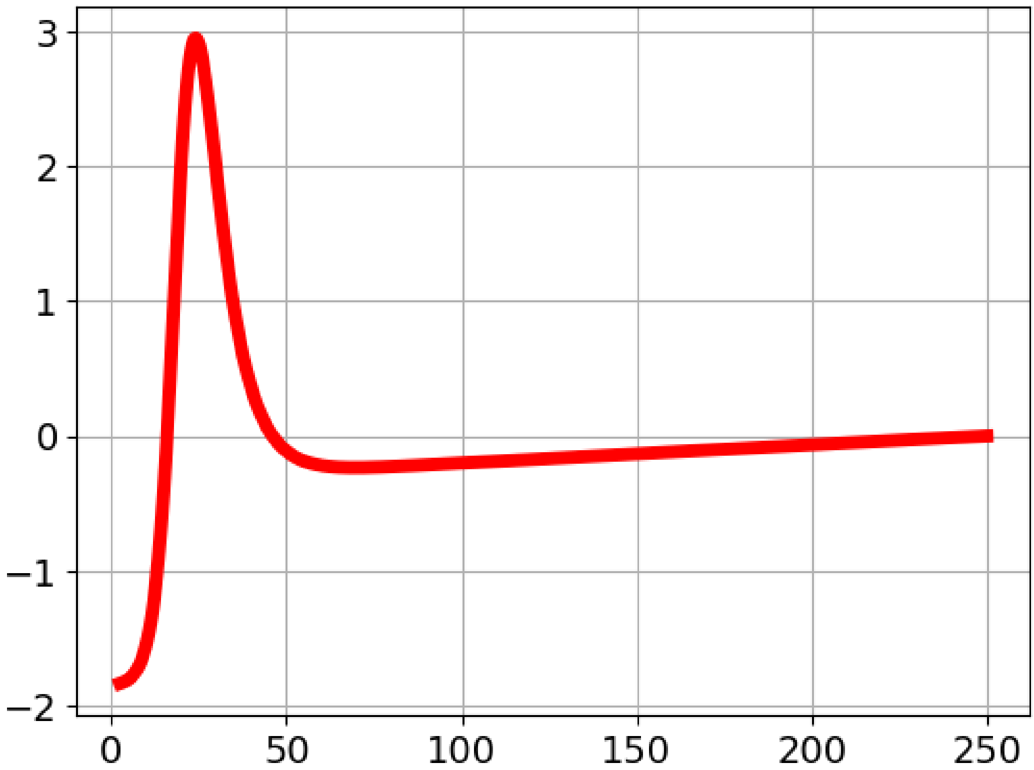
```
Guess = [guess]
print(Guess)
i3 = species.index('P3')

for g in Guess:
    x0[i3] = g

    ## OL simulate
    OLndat = st.
↪sim(s,OLsc,reduced=False,t=tt,X0=x0,parameter=parameter,quiet=True)
    dXP3 = OLndat['dX'][:,i3]
    v_ss = dXP3[-1]
    x_ss = OLndat['X'][-1,:]
    print('guess',g, ': v_ss',v_ss)

# print(x_ss)
print(species)
```

```
['A', 'G1_XM', 'G1_XP', 'G2_XM', 'G2_XP', 'G3_XM', 'G3_XP', 'P3']
[10.75855]
guess 10.75855 : v_ss -1.3583868704980517e-05
['A', 'G1_E', 'G1_EA', 'G1_M', 'P2', 'G1_EI', 'G1_EAI', 'R', 'G1_C0', 'G1_C1',
'G1_C2', 'G1_C3', 'G1_C4', 'G1_C5', 'G1_C6', 'G1_C7', 'G1_C8', 'P1', 'G1_XM',
'G1_XP', 'G2_E', 'G2_EA', 'G2_M', 'P3', 'G2_EI', 'G2_EAI', 'G2_C0', 'G2_C1',
'G2_C2', 'G2_C3', 'G2_C4', 'G2_C5', 'G2_C6', 'G2_C7', 'G2_C8', 'G2_XM', 'G2_XP',
'G3_E', 'G3_EA', 'G3_M', 'G3_EI', 'G3_EAI', 'G3_C0', 'G3_C1', 'G3_C2', 'G3_C3',
'G3_C4', 'G3_C5', 'G3_C6', 'G3_C7', 'G3_C8', 'G3_XM', 'G3_XP']
```

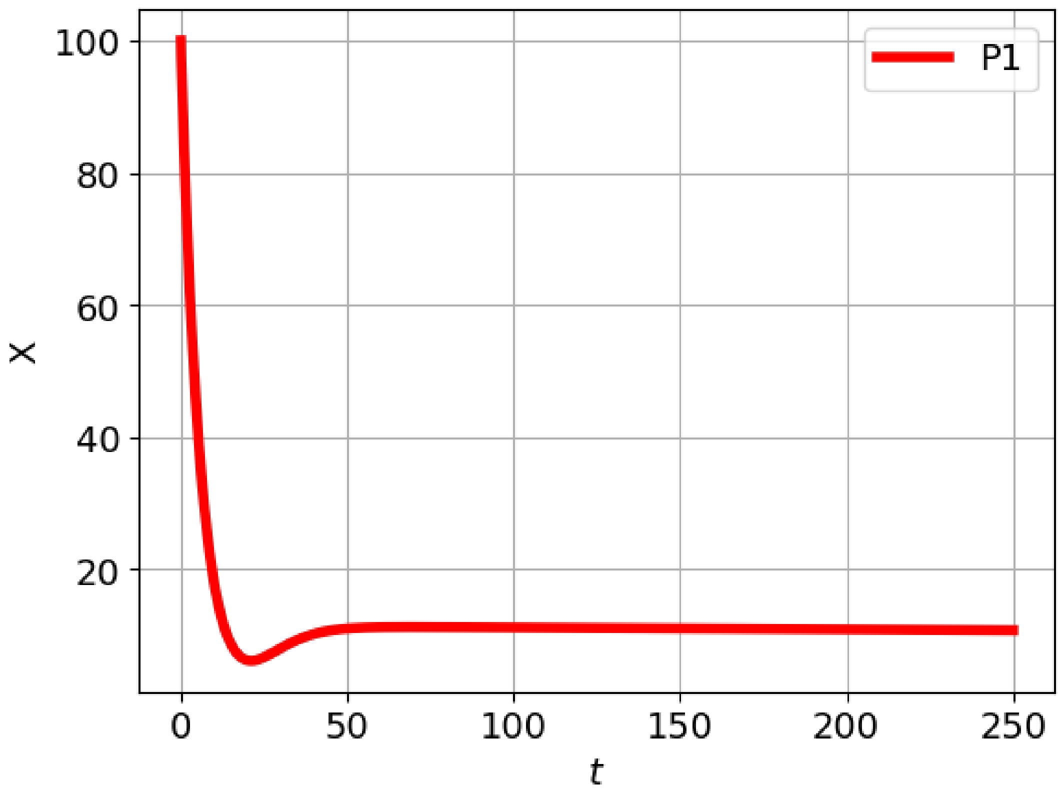
[49]:
```
# for i,spec in enumerate(species):
#     print(f"x_ss[species.index('{spec}')] = {x_ss[i]}")
```

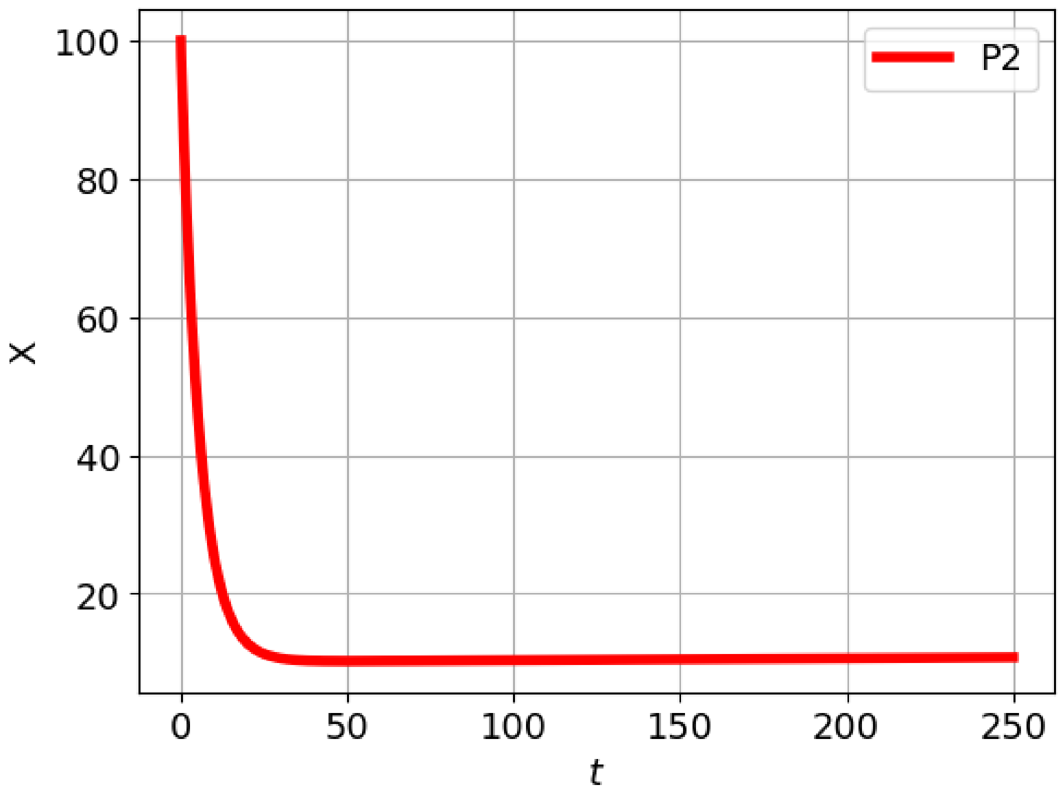
[50]:
```
plt.plot(tt[100:],dXP3[100:])
plt.grid()
```

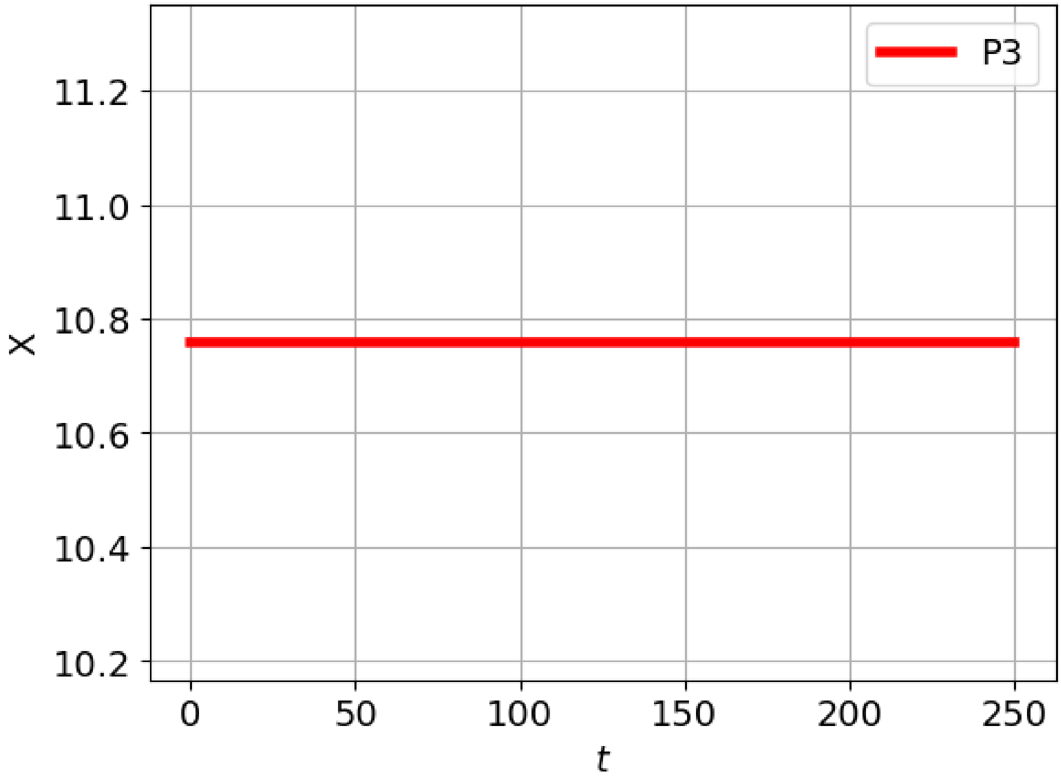
[50]:
```
[<matplotlib.lines.Line2D at 0x7253c5228a60>]
```

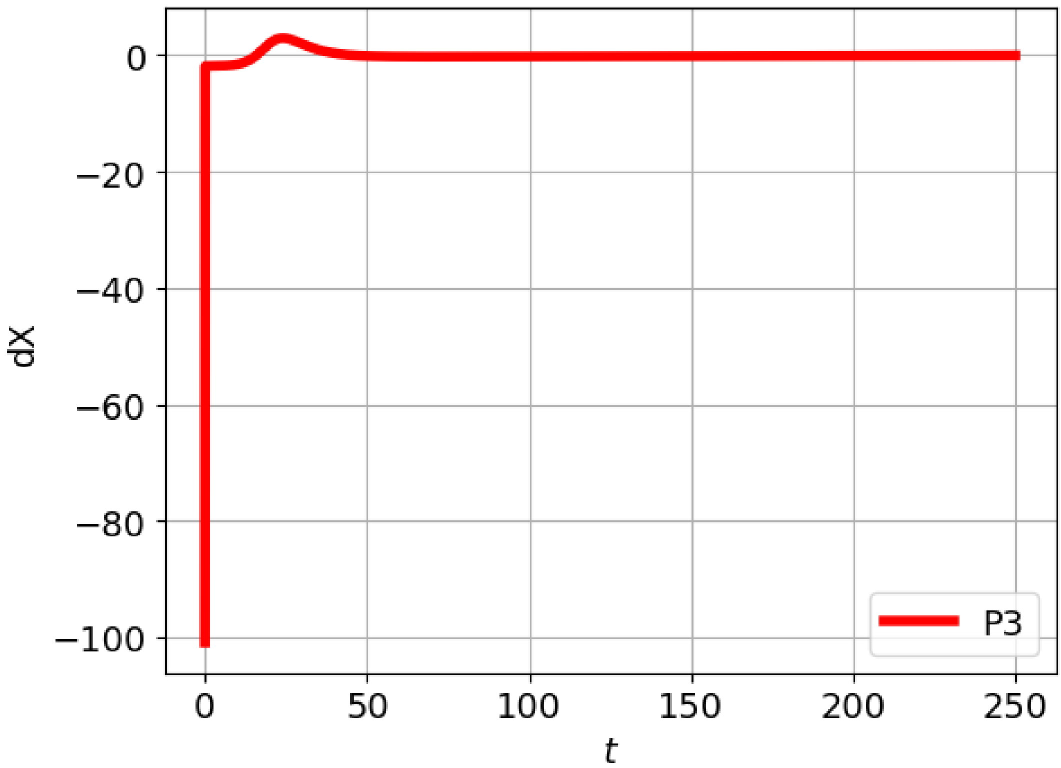


[51]:
```
st.plot(s,OLndat,species = ['P1'],reaction = [])
st.plot(s,OLndat,species = ['P2'],reaction = [])
st.plot(s,OLndat,species = ['P3'],reaction = [])
st.plot(s,OLndat,dX=True,species = ['P3'],reaction=[])
```


P1
100
80
60
40
20
X
0
50
100
150
200
250
t


P2
100
80
60
40
20
X
0
50
100
150
200
250
t


P3
11.2
11.0
10.8
10.6
10.4
10.2
X
0
50
100
150
200
250
t


0
−20
−40
−60
−80
−100
dX
P3
0
50
100
150
200
250
t

[52]:
```
## Refine
# x_ss,v_ss = SteadyState(s,OLsc,parameter,x0=x_ss)
# print(v_ss)
# x_ss,v_ss=findSteadyState(s,OLsc,parameter,x0)
```

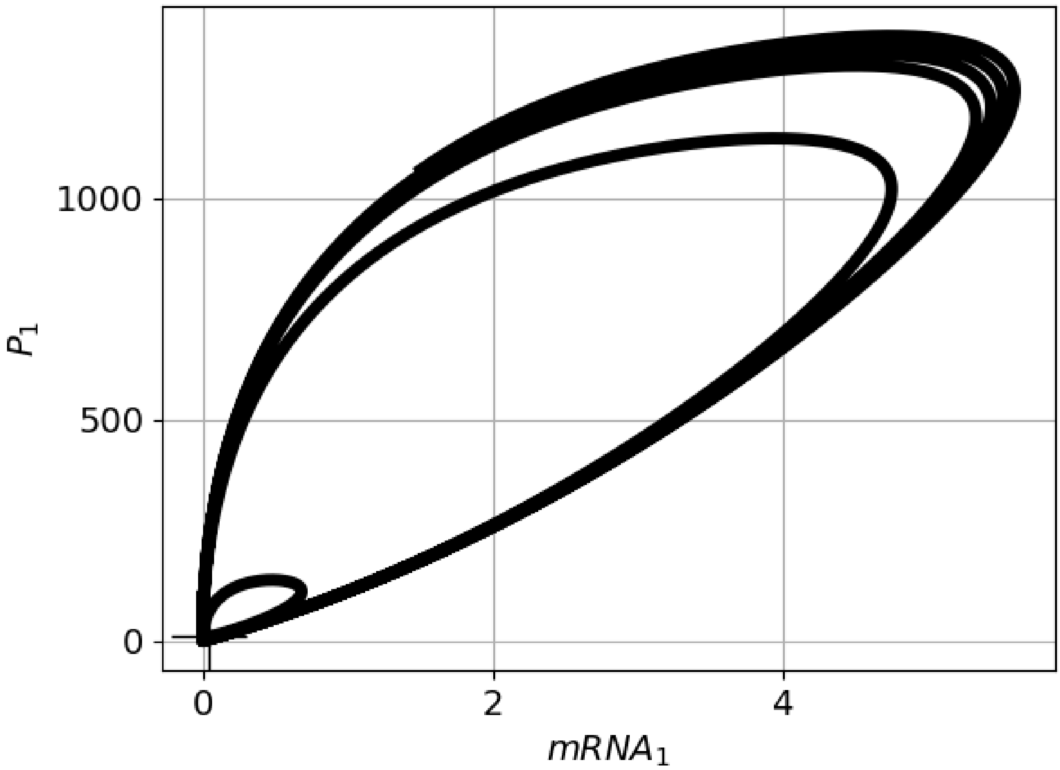
[53]:
```
## replot with x_ss
plt.plot(x[:,iM[0]],x[:,iP[0]],color='black')
plt.plot(x_ss[iM[0]],x_ss[iP[0]],marker='+',color='black')
plt.locator_params(nbins=4)
plt.grid()
plt.xlabel('$mRNA_1$')
plt.ylabel('$P_1$')
SaveFig(SystemName,'PhasePlane')
```

[53]: [<matplotlib.lines.Line2D at 0x7253c47aa340>]

[53]: [<matplotlib.lines.Line2D at 0x7253c47aa820>]

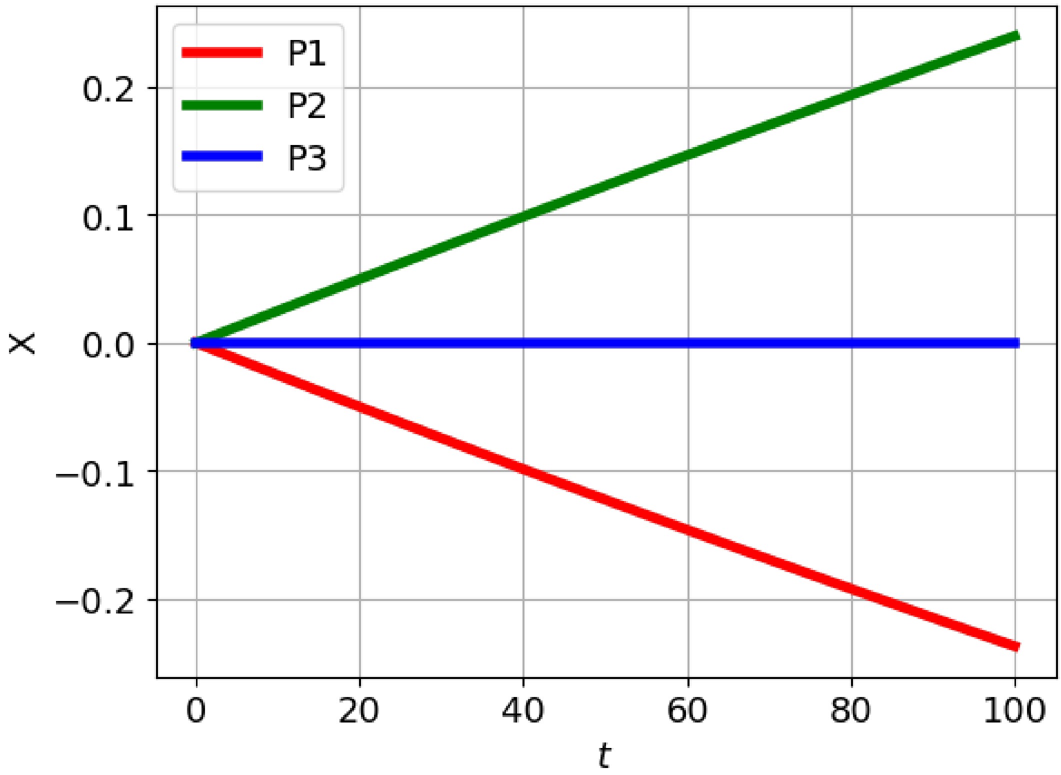
[53]: Text(0.5, 0, '$mRNA_1$')

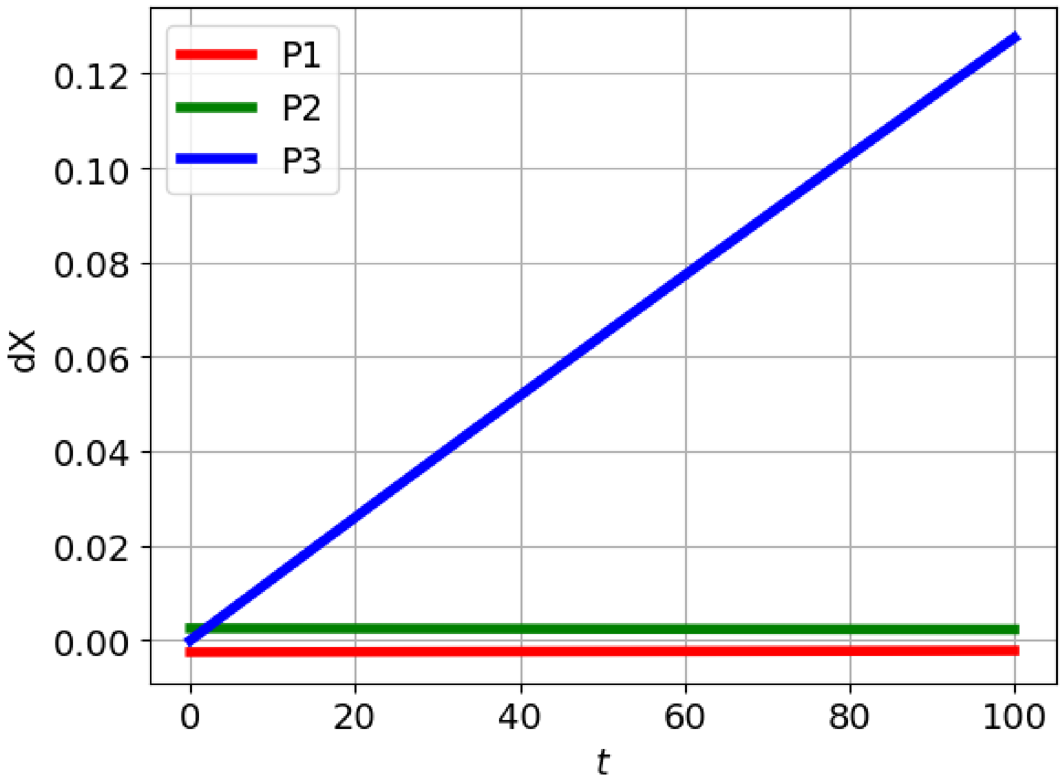
[53]: Text(0, 0.5, 'P_1')



[54]:
```
## Simulate from steady-state - just to check ...
ttt = np.linspace(0,100)
OLndat_ss = st.
↪sim(s,OLsc,reduced=False,t=ttt,X0=x_ss,parameter=parameter,quiet=True)
```

[55]:
```
st.plot(s,OLndat_ss,x_ss=x_ss,species=['P1','P2','P3'],reaction=[])
st.plot(s,OLndat_ss,dX=True,species=['P1','P2','P3'],reaction=[])
```





15 Simulate from perturbed steady-state

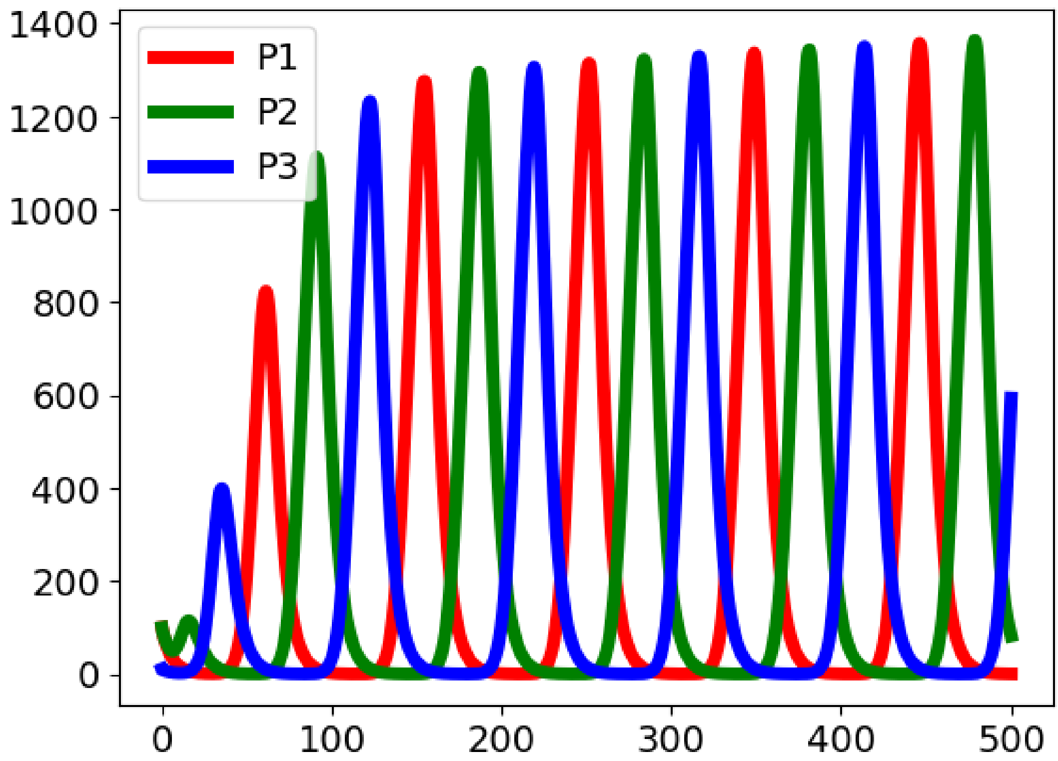
[56]:
```
timespan = [0.0, 500.0]
t_sim, x = bgt.simulate(model, timespan=timespan, x0=x0, dt=1e-3)
```

[57]:
```
# Protein amounts
plt.figure()
## Proteins
iP = [species.index('P1'), species.index('P2'), species.index('P3')]
## mRNA
iM = [species.index('G1_M'),species.index('G2_M'),species.index('G3_M')]
plt.plot(t_sim,x[:,iP])
plt.legend(['P1','P2','P3'])
plt.show()
plt.plot(t_sim,x[:,iM])
plt.legend(['M1','M2','M3'])
plt.show()
```

[57]:
```
<Figure size 640x480 with 0 Axes>
```

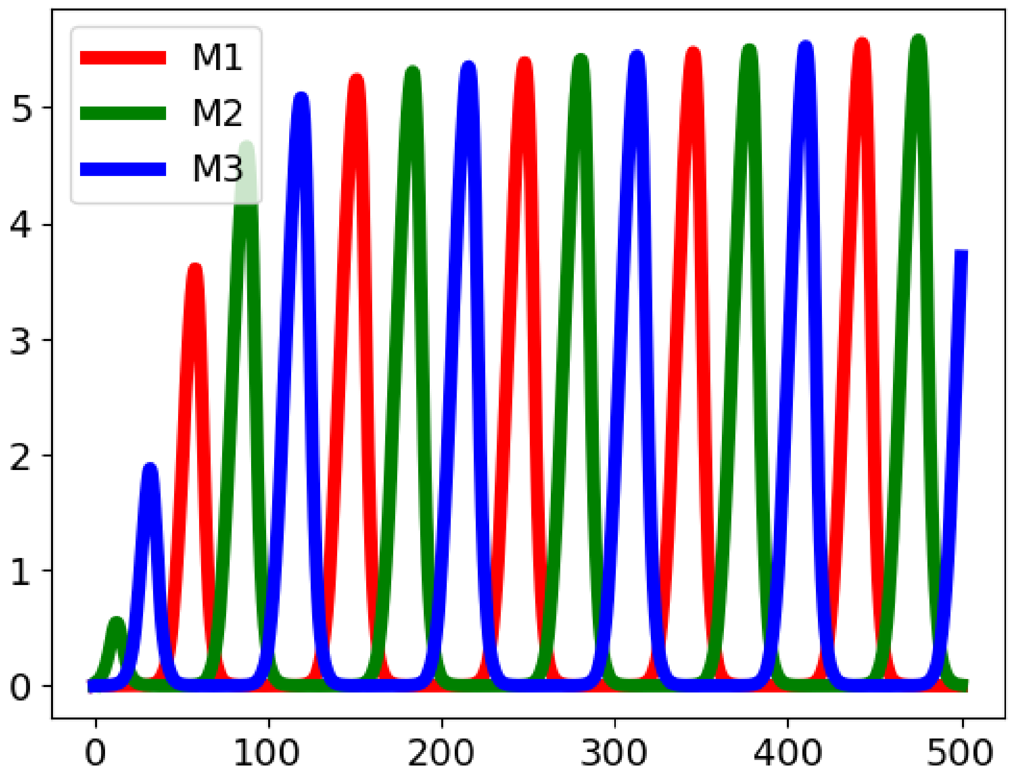
[57]:
```
[<matplotlib.lines.Line2D at 0x7253a3b48550>,
 <matplotlib.lines.Line2D at 0x7253a3b48b50>,
 <matplotlib.lines.Line2D at 0x7253a3b487f0>]
```

[57]:
```
<matplotlib.legend.Legend at 0x7253c4256df0>
```



[57]: [<matplotlib.lines.Line2D at 0x7253a36bb340>,
 <matplotlib.lines.Line2D at 0x7253a36bb3a0>,
 <matplotlib.lines.Line2D at 0x7253a36bb3d0>]

[57]: <matplotlib.legend.Legend at 0x7253a3520790>



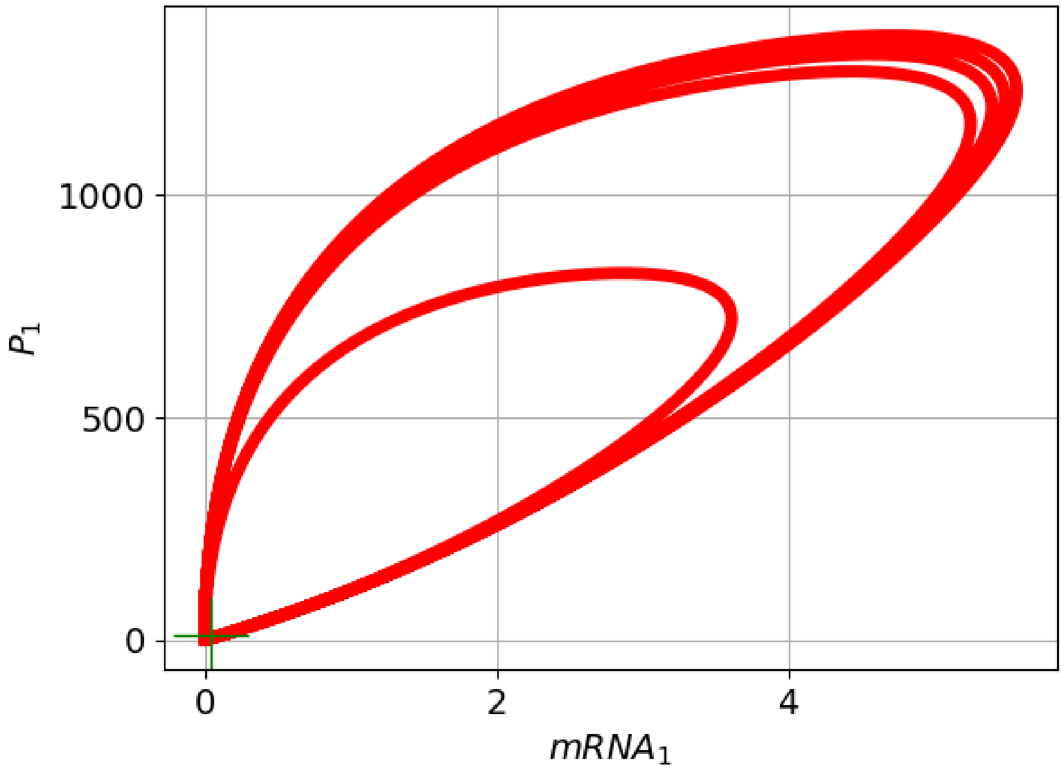
[58]:
```
plt.plot(x[:,iM[0]],x[:,iP[0]])
plt.grid()
plt.xlabel('$mRNA_1$')
plt.ylabel('$P_1$')
plt.plot(x_ss[iM[0]],x_ss[iP[0]],marker='+')
plt.locator_params(nbins=4)

# SaveFig(SystemName,'PhasePlane')
```

[58]: [<matplotlib.lines.Line2D at 0x7253a3b35e80>]

[58]: Text(0.5, 0, '$mRNA_1$')

[58]: Text(0, 0.5, 'P_1')

[58]: [<matplotlib.lines.Line2D at 0x7253c45df460>]



16 Linearise

[59]:
```
Sys = st.lin(s,OLsc,parameter=parameter,x_ss=x_ss,outvar='dX',quiet=True)
```

[60]:
```
## TF from x_P to dx_P
sys = ExtractSubsystem(Sys,OLchemostats.index('P3'),species.index('P3'))
L0 = -sys

## Tf from x_A to dx_P
sys_F = ExtractSubsystem(Sys,OLchemostats.index('A'),species.index('P3'))
F0 = sys_F

# print(np.sort(con.poles(sys)))

## Reduced-order subsystems

print('L0_r:')
L0_r = con.balred(L0,redOrder)
L0_lr = con.balred(L0,largeOrder) #make it numerically managable
con.tf(L0_r)

pole_r,zero_r = con.pzmap(L0_r,plot=None)
for i in range(redOrder):
    print(f'Pole:{pole_r[i]:.2e}\t Zero:{zero_r[i]:.2e}')
```

```python
print('F0_r:')
F0_r = con.balred(F0,redOrder)
con.tf(F0_r)

## System + integrator
L_r = IntegrateTF(L0_r)
L = IntegrateTF(L0_lr)  #make it numerically managable
# con.tf(L_r)
```

```
3 states have been removed from the model
3 states have been removed from the model
L0_r:
```

```
/home/peterg/anaconda3/envs/bgt/lib/python3.8/site-
packages/slycot/exceptions.py:241: SlycotResultWarning:
The selected order 30 is greater
than the order of a minimal realization of the
given system. `nr` was set automatically to 15
corresponding to the order of a minimal realization
of the system
  warn(globals()[warning](fmessage, iwarn, info))
```

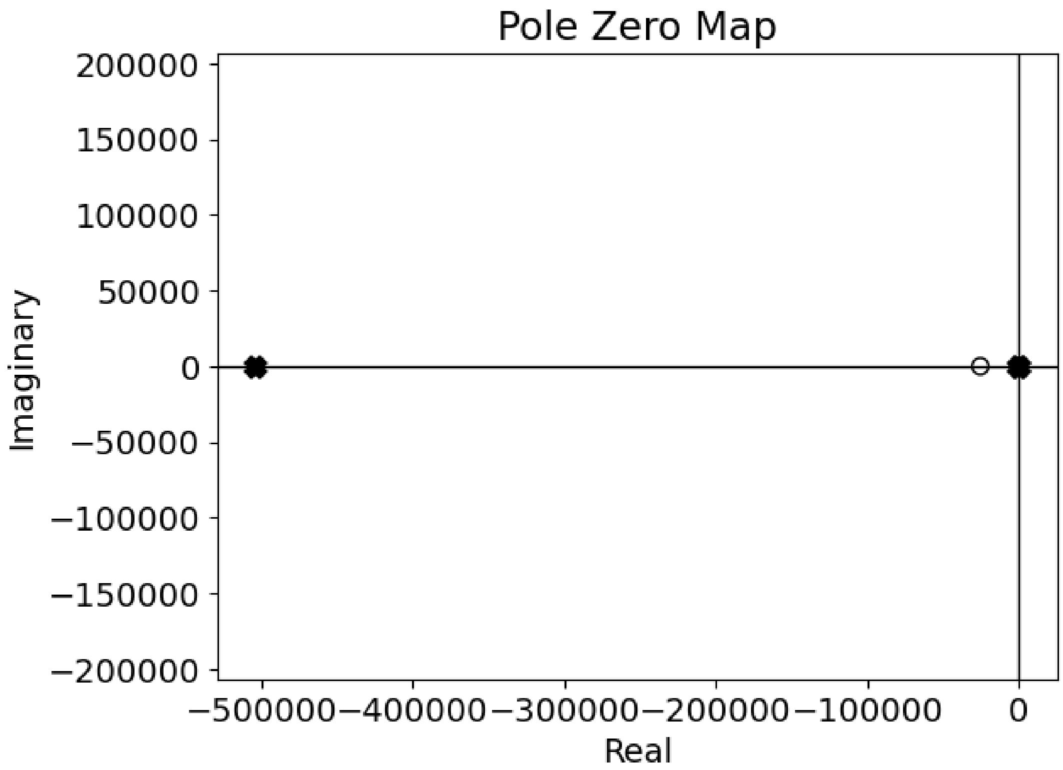
[60]:

$$\frac{5.469(s-(0.05911+0.2379j))(s-(0.05911-0.2379j))(s+2.535\times 10^4)}{(s+(0.08214-0.06437j))(s+(0.08214+0.06437j))(s+5.04\times 10^5)}$$

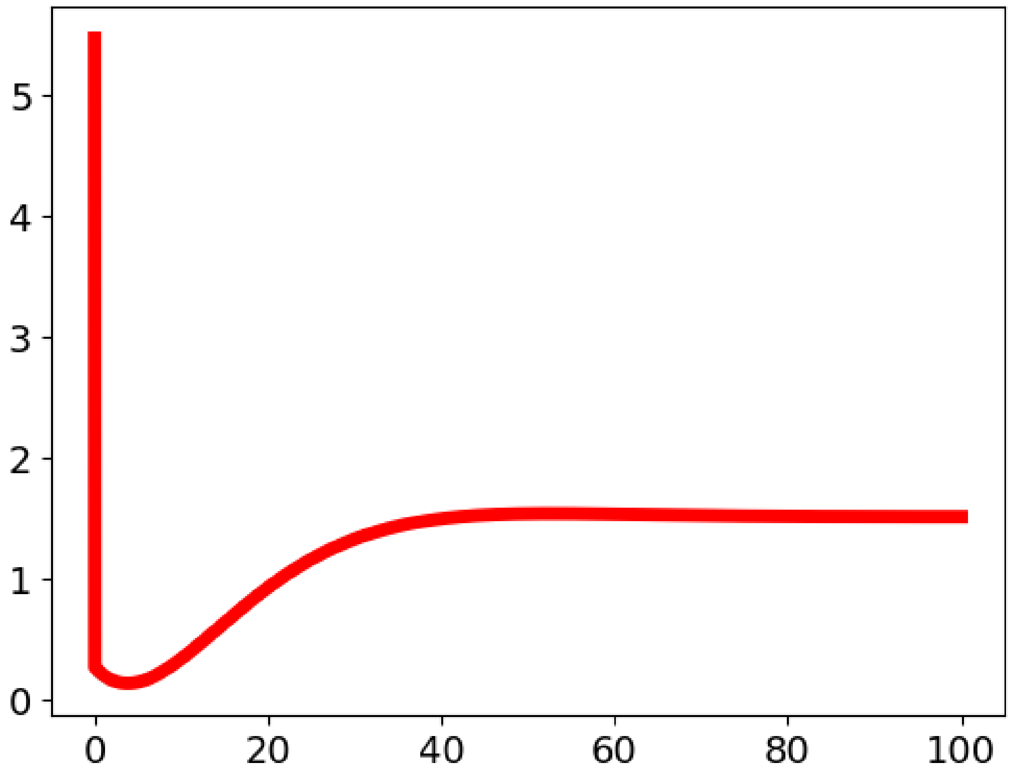
```
Pole:-5.04e+05+0.00e+00j          Zero:-2.54e+04+0.00e+00j
Pole:-8.21e-02+6.44e-02j          Zero:5.91e-02+2.38e-01j
Pole:-8.21e-02-6.44e-02j          Zero:5.91e-02-2.38e-01j
F0_r:
```

[60]:

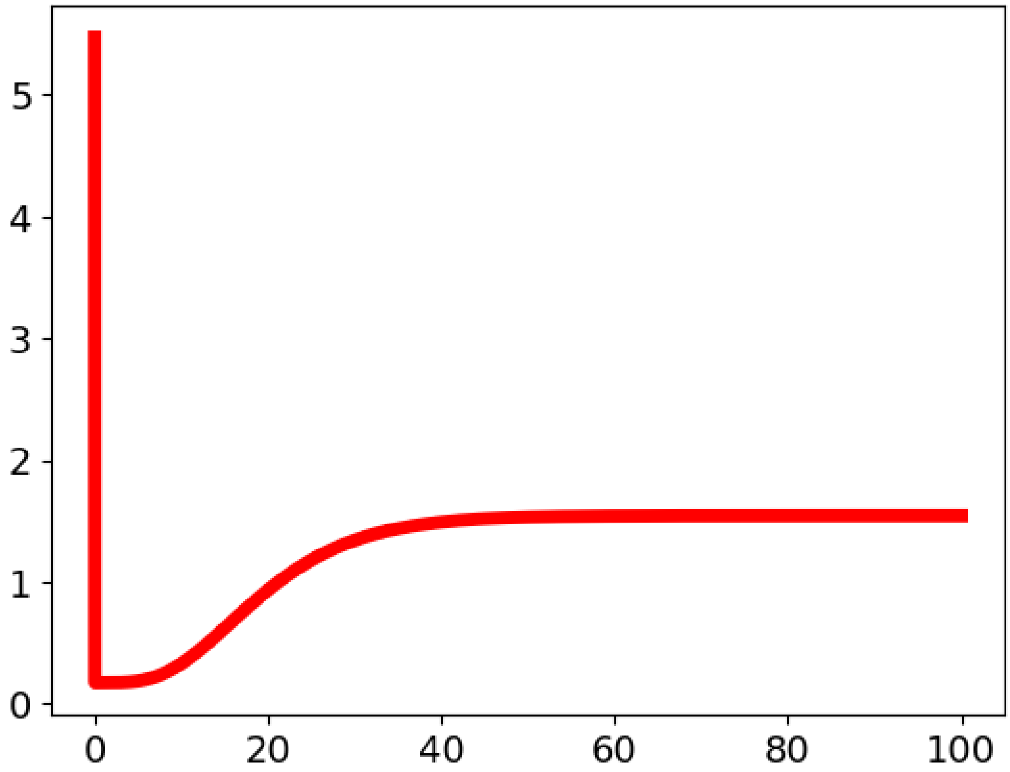
$$\frac{0.9396(s+270.2)(s+999.7)}{(s+(484.5-1026j))(s+(484.5+1026j))(s+5.045\times 10^5)}$$



[61]:
```
t_ol = np.linspace(0,100,10000)
y_r = step_response(L0_r,T=t_ol)
plt.show()
y = step_response(L0,T=t_ol)
plt.show()
plt.plot(t_ol,y_r,label='y_r')
plt.plot(t_ol,y,label='y')
plt.legend()
```



5
4
3
2
1
0
0
20
40
60
80
100

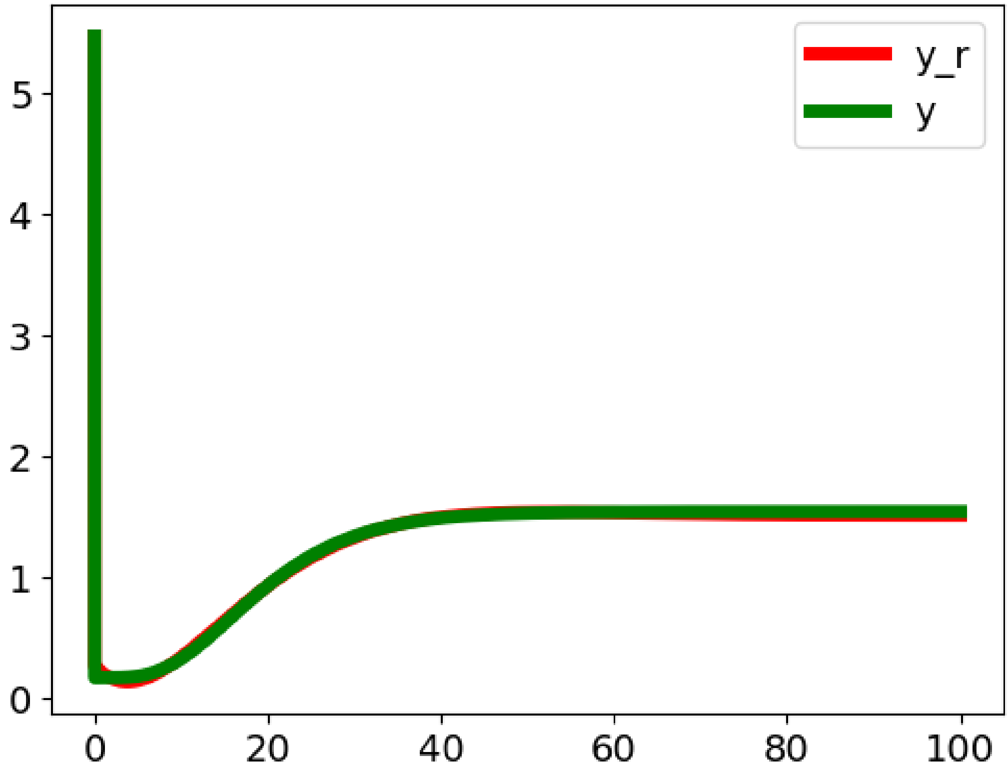


5
4
3
2
1
0
0
20
40
60
80
100

[61]: [<matplotlib.lines.Line2D at 0x7253c4424fd0>]

[61]: [<matplotlib.lines.Line2D at 0x7253c441e310>]

[61]: <matplotlib.legend.Legend at 0x7253c44214f0>

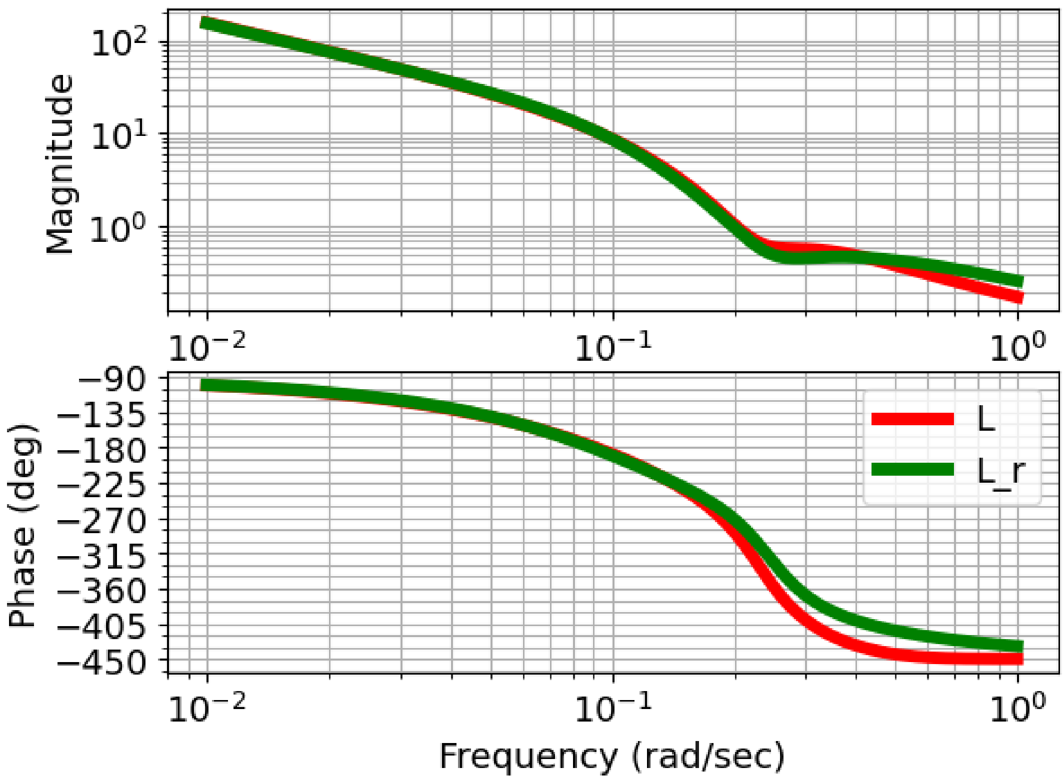


17 Bode Plots of loop-gain

[62]:
```
Omega = np.logspace(-2,0,500)
mag,phase,omega = con.bode_plot([L,L_r],omega=Omega)
plt.legend(['L','L_r'])
gm, pm, wcg, wcp = con.margin(L)
print(f'Phase margin = {pm:.0f} deg at w = {wcp:.4f}, f = {wcp/(2*np.pi):.4}␣
↪Hz, period = {2*np.pi/wcp:0.4}')
print(f'Gain margin = {gm:.4f} at f = {wcg/(2*np.pi):.4} Hz')

print(f'$\\theta_{{pm}} = {round(pm)}^\\circ$')
print(f'$\\omega_{{pm}} = \\SI{{{wcp:.2f}}}{{\\radian\\per\\second}}$')
```

[62]: <matplotlib.legend.Legend at 0x7253a3cfb7f0>

```
Phase margin = -111 deg at w = 0.2026, f = 0.03225 Hz, period = 31.01
Gain margin = 0.0953 at f = 0.01449 Hz
$\theta_{pm} = -111^\circ$
$\omega_{pm} = \SI{0.20}{\radian\per\second}$
```



17.1 Linear closed-loop

[63]:
```
Integrator = con.tf(1,[1,0])
CLsys = con.feedback(Integrator,L0)
CLsys_r = con.feedback(Integrator,L0_r)
# CLsys = con.minreal(con.series(F0,CLsys0))
# CLsys_r = con.minreal(con.series(F0_r,CLsys0_r))

con.tf(CLsys_r)
poles = con.poles(CLsys_r)
print(poles)
```

[63]:

$$\frac{(s + (0.08214 - 0.06437j))(s + (0.08214 + 0.06437j))(s + 5.04 \times 10^5)}{(s - (0.0488 + 0.1685j))(s - (0.0488 - 0.1685j))(s + 0.537)(s + 5.04 \times 10^5)}$$

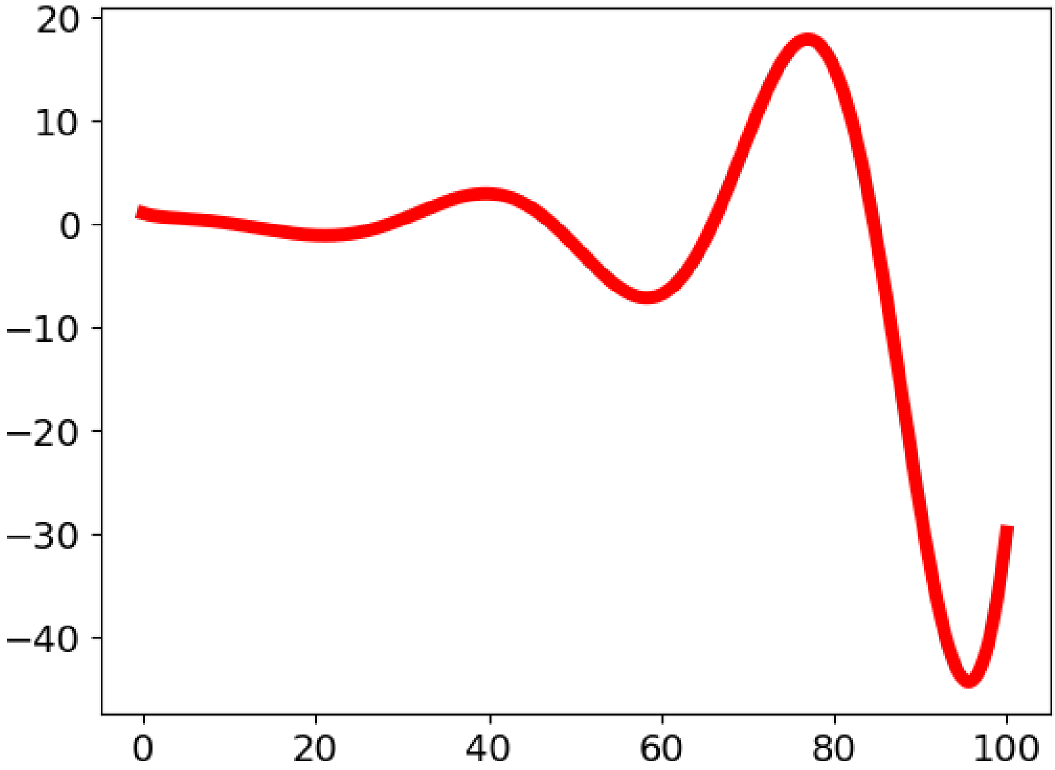
```
[-5.03998114e+05+0.j         -5.36958776e-01+0.j
  4.87967143e-02+0.16850036j  4.87967143e-02-0.16850036j]
```

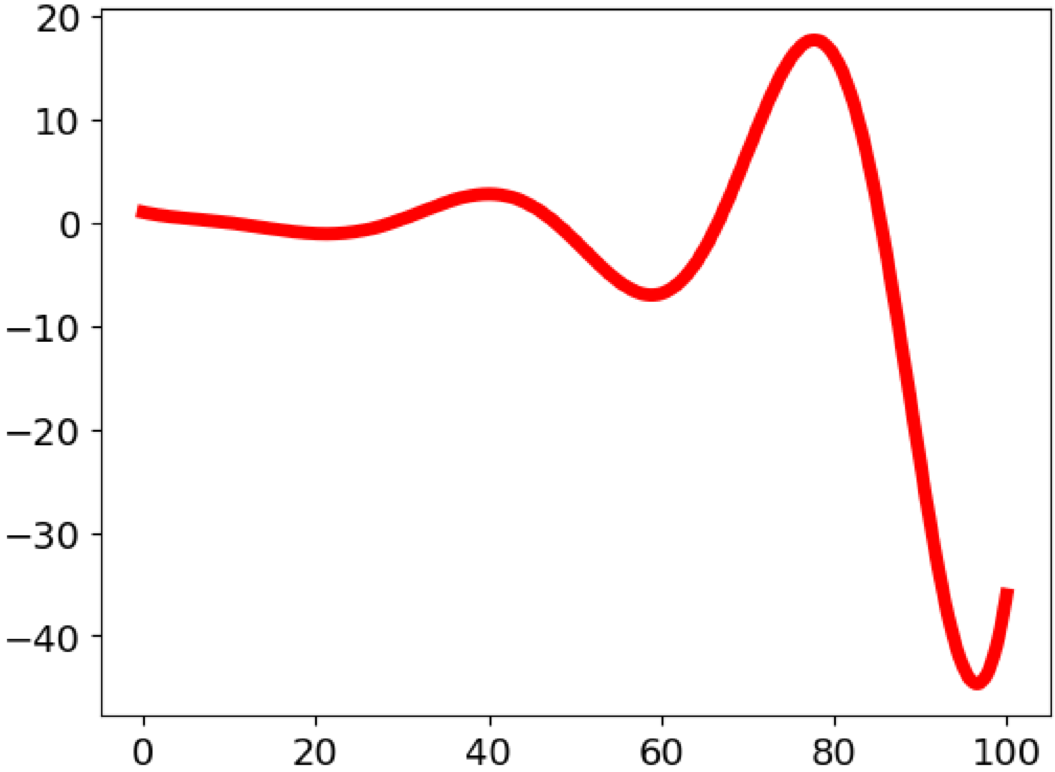
[64]:
```
imag = np.imag(poles[2])
print(imag)
Freq = imag/(2*np.pi)
print(Freq)
print(1/Freq)
```

```
0.1685003566224878
0.026817664669216118
```

37.28885465362298

[65]:
```
y_r = impulse_response(CLsys_r,T=t_ol)
plt.show()
y = impulse_response(CLsys,T=t_ol)
plt.show()
plt.plot(t_ol,y_r,label='y_r')
plt.plot(t_ol,y,label='y')
plt.legend()
```

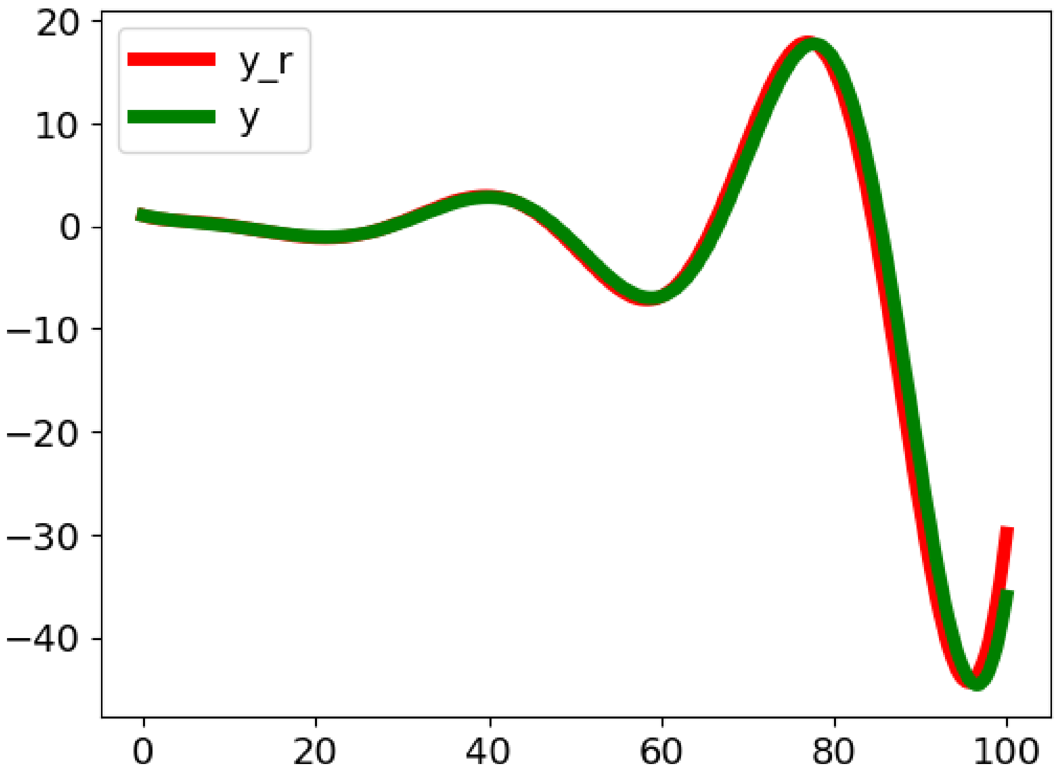




[65]: [<matplotlib.lines.Line2D at 0x7253a2dbf6a0>]

[65]: [<matplotlib.lines.Line2D at 0x7253a2dbfc70>]

[65]: <matplotlib.legend.Legend at 0x7253a2dcfc10>



17.2 Non-linear simulation

[66]:
```
## Simulate
X0 = copy.copy(x_ss)
pert = 0.1
X0[i3] += pert

# t_sim  =  np.linspace(0,130,10000)
t_sim = t_ol
ndat = st.sim(s,sc=sc,t=t_sim,X0=X0,parameter=parameter,quiet=True)
```

[67]:
```
y_n = (ndat['X'][:,i3] - x_ss[i3] )
plt.plot(t_sim,y_n, label='nlin',lw=5)
plt.plot(t_ol,pert*y,label='lin',lw=5)
plt.plot(t_ol,pert*y_r,label='lin (reduced)',lw=5,ls='dashed')
plt.grid()
plt.legend()
plt.xlabel('$t$')
plt.ylabel('$x_P-x_{ss}$')
# plt.xlim(right=max(t_ol))
SaveFig(SystemName,'Simulation')
```

[67]:
```
[<matplotlib.lines.Line2D at 0x7253a2d98a00>]
```

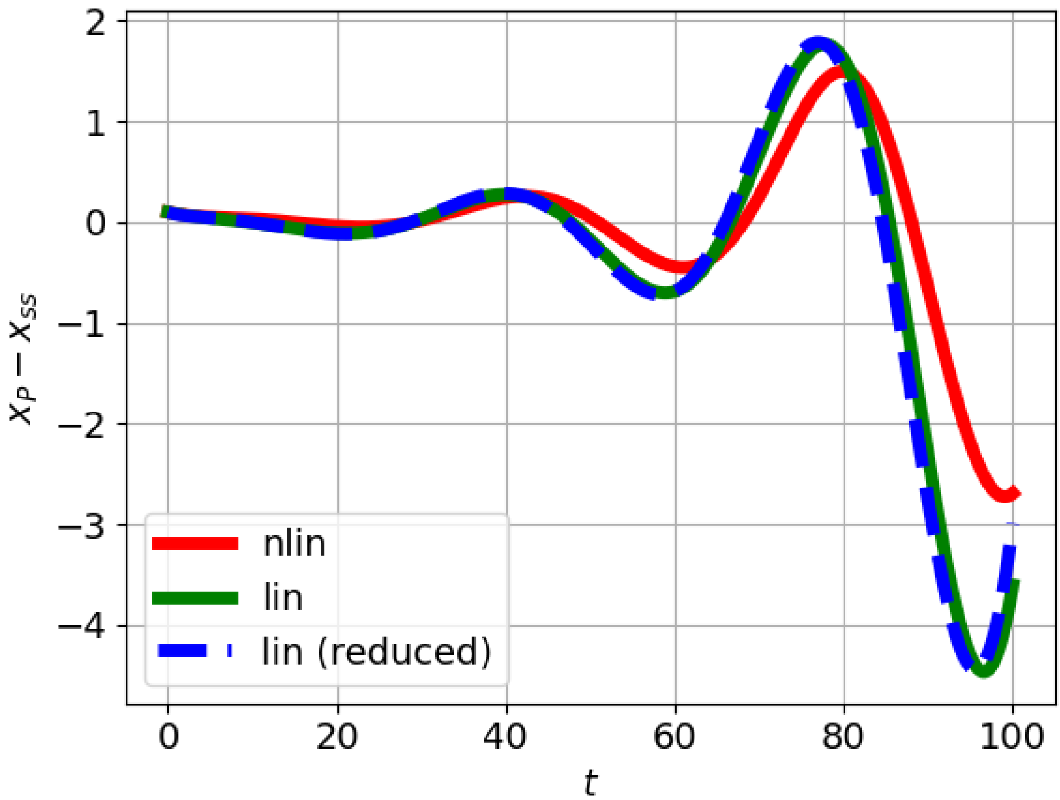
[67]:
```
[<matplotlib.lines.Line2D at 0x7253a2d98d60>]
```

[67]: [<matplotlib.lines.Line2D at 0x7253a2d98f10>]

[67]: <matplotlib.legend.Legend at 0x7253a2da6190>

[67]: Text(0.5, 0, 't')

[67]: Text(0, 0.5, 'x_P-x_{ss}')



17.3 Period

[68]:
```
# i_zc = zero_crossings(y_n)

# t_zc = t_sim[i_zc].T
# T_zc = np.diff(t_zc)
# print(t_zc)
# print(T_zc)

# plt.hlines([1/
↪Freq],min(t_zc),max(t_zc),ls='dashed',color='grey',label='linear')
# plt.plot(t_zc[1:],T_zc, label='actual')

# plt.grid()
# plt.legend()
# plt.xlabel('$t$')
# plt.ylabel('$T$')
```

```
# SaveFig(SystemName,'Period')

# # print(zc)
# print(1/Freq)
```

18 Split-loop model

[69]:
```
OutpVar = 'P3'
SplitVar = 'P3f'
```

[70]:
```
## Modify orignal list of reactions to insert split-loop variable SplitVar
SLreactions = []
for reac in reactions:
    react = reac[0]
    name = reac[1]
    if '2*P3' in react:
        print(f'Reaction {name} is {react}')
        react = react.replace(OutpVar,SplitVar)
        print(f'Reaction {name} is now {react}')
    SLreactions.append((react,name))

SLrn = Reaction_Network(name="RepressilatorSL")
for (reaction_string,reaction_name) in SLreactions:
    # print(reaction_name,reaction_string)
    SLrn.add_reaction(reaction_string, name=reaction_name)
```

```
Reaction G2_Tc3 is G2_E + 2*P3 = G2_EI
Reaction G2_Tc3 is now G2_E + 2*P3f = G2_EI
Reaction G2_Tc4 is G2_EA + 2*P3 = G2_EAI
Reaction G2_Tc4 is now G2_EA + 2*P3f = G2_EAI
```

[ ]:
```

```

[71]:
```
SLs0 = rn2bg(SLrn,'RepressilatorSL_abg')
import RepressilatorSL_abg
imp.reload(RepressilatorSL_abg)
```

[71]:
```
<module 'RepressilatorSL_abg' from '/home/peterg/WORK/Research/SystemsBiology/
↪No
tes/2024/Repressilator/RepressilatorSL_abg.py'>
```

18.1 Stoichiometry

[72]:
```
SLs = st.stoich(RepressilatorSL_abg.model(),quiet=True)
species_sl = SLs['species']
```

[73]:
```
for spec in SLs['species']:
    if spec[0]=='P':
        print(spec)
```

```
P2
P1
P3f
P3
```

[74]:

```
## Add SplitVar to the OL chemostat list.
SLchemostats = copy.copy(OLchemostats)
SLchemostats.append(SplitVar)
print(SLchemostats)
SLsc = st.statify(SLs,chemostats=SLchemostats)
```

```
['A', 'G1_XM', 'G1_XP', 'G2_XM', 'G2_XP', 'G3_XM', 'G3_XP', 'P3', 'P3f']
```

[75]:

```
## Reactions
disp.Latex(st.sprintrl(SLs,chemformula=False,split=16,all=True))
```

[75]:

$$A + G1E \Leftrightarrow G1EA \tag{143}$$

$$G1EA \Leftrightarrow G1E + G1M \tag{144}$$

$$G1E + 2P2 \Leftrightarrow G1EI \tag{145}$$

$$G1EA + 2P2 \Leftrightarrow G1EAI \tag{146}$$

$$G1M + R \Leftrightarrow G1C0 \tag{147}$$

$$A + G1C0 \Leftrightarrow G1M + G1C1 \tag{148}$$

$$A + G1C1 \Leftrightarrow G1C2 \tag{149}$$

$$A + G1C2 \Leftrightarrow G1C3 \tag{150}$$

$$A + G1C3 \Leftrightarrow G1C4 \tag{151}$$

$$A + G1C4 \Leftrightarrow G1C5 \tag{152}$$

$$A + G1C5 \Leftrightarrow G1C6 \tag{153}$$

$$A + G1C6 \Leftrightarrow G1C7 \tag{154}$$

$$A + G1C7 \Leftrightarrow G1C8 \tag{155}$$

$$G1C8 \Leftrightarrow R + P1 \tag{156}$$

$$G1M \Leftrightarrow G1XM \tag{157}$$

$$P1 \Leftrightarrow G1XP \tag{158}$$

$$A + G2E \Leftrightarrow G2EA \tag{159}$$
$$G2EA \Leftrightarrow G2E + G2M \tag{160}$$
$$G2E + 2P3f \Leftrightarrow G2EI \tag{161}$$
$$G2EA + 2P3f \Leftrightarrow G2EAI \tag{162}$$
$$R + G2M \Leftrightarrow G2C0 \tag{163}$$
$$A + G2C0 \Leftrightarrow G2M + G2C1 \tag{164}$$
$$A + G2C1 \Leftrightarrow G2C2 \tag{165}$$
$$A + G2C2 \Leftrightarrow G2C3 \tag{166}$$
$$A + G2C3 \Leftrightarrow G2C4 \tag{167}$$
$$A + G2C4 \Leftrightarrow G2C5 \tag{168}$$
$$A + G2C5 \Leftrightarrow G2C6 \tag{169}$$
$$A + G2C6 \Leftrightarrow G2C7 \tag{170}$$
$$A + G2C7 \Leftrightarrow G2C8 \tag{171}$$
$$G2C8 \Leftrightarrow P2 + R \tag{172}$$
$$G2M \Leftrightarrow G2XM \tag{173}$$
$$P2 \Leftrightarrow G2XP \tag{174}$$

$$A + G3E \Leftrightarrow G3EA \tag{175}$$
$$G3EA \Leftrightarrow G3E + G3M \tag{176}$$
$$2P1 + G3E \Leftrightarrow G3EI \tag{177}$$
$$2P1 + G3EA \Leftrightarrow G3EAI \tag{178}$$
$$R + G3M \Leftrightarrow G3C0 \tag{179}$$
$$A + G3C0 \Leftrightarrow G3M + G3C1 \tag{180}$$
$$A + G3C1 \Leftrightarrow G3C2 \tag{181}$$
$$A + G3C2 \Leftrightarrow G3C3 \tag{182}$$
$$A + G3C3 \Leftrightarrow G3C4 \tag{183}$$
$$A + G3C4 \Leftrightarrow G3C5 \tag{184}$$
$$A + G3C5 \Leftrightarrow G3C6 \tag{185}$$
$$A + G3C6 \Leftrightarrow G3C7 \tag{186}$$
$$A + G3C7 \Leftrightarrow G3C8 \tag{187}$$
$$G3C8 \Leftrightarrow R + P3 \tag{188}$$
$$G3M \Leftrightarrow G3XM \tag{189}$$
$$P3 \Leftrightarrow G3XP \tag{190}$$

[76]:
```
species_sl = SLs['species']
species = s['species']
# print(len(species_sl),species_sl)
```

18.2 Steady-state analysis

[77]:
```
## Create the steady state corresponding to open loop with x_inh=x_P:S
x_sl_ss = np.ones(SLs['n_X'])
X_ss = copy.copy(x_ss)
for i,spec in enumerate(species):
    # print(spec)
    x_sl_ss[species_sl.index(spec)] = X_ss[i]
x_sl_ss[species_sl.index(SplitVar)] = X_ss[species.index(OutpVar)]

## Make parameters the same
K_out = parameter['K_'+OutpVar]
# print(K_out)
parameter['K_'+SplitVar] = K_out
```

[78]:
```
# for key in parameter:
#     if key in ['K_P3f']:
#         print('****')
#     print(key, parameter[key])
```

18.3 Linearise

[79]:
```
Sys = st.lin(SLs,SLsc,parameter=parameter,x_ss=x_sl_ss,outvar='dX',quiet=True)
```

[80]:
```
Inp = [SplitVar,OutpVar]
Outp = Inp

TF = {}
for inp in Inp:
    for outp in Outp:
        key = inp+'_'+outp
        print(key)
        sys = ExtractSubsystem(Sys,SLchemostats.index(inp),species_sl.
 ↪index(outp))
        degree = len(sys.B)
        print(degree)
        # maxdegree = 4
        print('gain',con.dcgain(sys))
        if degree>redOrder:
            sys_r = con.balred(sys,redOrder)
        else:
            sys_r = sys
        sys_r = con.minreal(sys_r,tol = 1e-6)

        print('gain_r',con.dcgain(sys_r))
        con.tf(sys_r)
        print(con.poles(sys_r))
        TF[key] =   con.tf(sys_r)
```

```
P3f_P3f
39 states have been removed from the model
2
```

```
gain -7.202571872255703e-15
0 states have been removed from the model
gain_r 9.202361095361766e-13
```

[80]:

$$\frac{-0.1019(s - 2.262 \times 10^{-9})(s + 1.497 \times 10^{10})}{(s + 87.72)(s + 1.497 \times 10^{10})}$$

```
[-1.49684864e+10+0.j -8.77191299e+01+0.j]
P3f_P3
3 states have been removed from the model
38
gain -1.3731107731905767
0 states have been removed from the model
gain_r -1.3875522550223096
```

[80]:

$$\frac{-0.005713(s - (0.6279 + 0.494j))(s - (0.6279 - 0.494j))}{(s + 0.08603)(s + (0.1119 - 0.1342j))(s + (0.1119 + 0.1342j))}$$

```
[-0.08602848+0.j         -0.11193919+0.13422377j -0.11193919-0.13422377j]
P3_P3f
5 states have been removed from the model
36
gain -7.622218992898018e-17
3 states have been removed from the model
gain_r 0.0

/home/peterg/anaconda3/envs/bgt/lib/python3.8/site-
packages/scipy/signal/_filter_design.py:1746: BadCoefficients: Badly conditioned
filter coefficients (numerator): the results may be meaningless
  warnings.warn("Badly conditioned filter coefficients (numerator): the "
/home/peterg/anaconda3/envs/bgt/lib/python3.8/site-
packages/scipy/signal/_filter_design.py:1091: RuntimeWarning: invalid value
encountered in divide
  b /= b[0]
```

[80]:

$$\frac{0}{1}$$

```
[]
P3_P3
4 states have been removed from the model
37
gain -0.17328680660197282
2 states have been removed from the model
gain_r -0.1732867955062929
```

[80]:

$$\frac{-5.367(s + 1.627 \times 10^{4})}{s + 5.04 \times 10^{5}}$$

```
[-504000.0112114+0.j]
```

18.4 Active and passive loop gains

[81]:
```
# print(f'Gain L0 = {con.dcgain(L0):.2f}')
# L0_pas_P = TF['P_P']
# K =  parameter['K_decr3_A']*parameter['kappa_decr3_r']
# K_f =  parameter['kappa_decr3_rf']
# print(f'Gain L0_pas_P = {con.dcgain(L0_pas_P):0.2f} ({K_f:0.2f})')
```

[82]:
```
LL0 = con.tf(0,1)
for index in TF:
    LL0 = con.parallel(LL0,-TF[index])

LL = IntegrateTF(LL0)

LL0_r = con.tf(con.balred(con.ss(LL0),redOrder))
LL_r = IntegrateTF(LL0_r)

## Check the same as derived from OL analysis
print('LL0_r (from open-loop')
con.tf(LL0_r)
print('L0_r (from split-loop)')
con.tf(L0_r)


L0_act = -TF[SplitVar+'_'+OutpVar]
L_act = IntegrateTF(L0_act)
L0_act_r = con.tf(con.balred(con.ss(L0_act),redOrder))
print(f'L0_act_r: ({con.dcgain(L0_act_r):0.2f})')
con.tf(L0_act_r)

L0_pas = con.minreal(con.
 ↪parallel(-TF[OutpVar+'_'+OutpVar],-TF[SplitVar+'_'+SplitVar]),tol=1e-3)
print('L0_pas')
print(f'L0_pas: ({con.dcgain(L0_pas):0.2f})')
con.tf(L0_pas)
L_pas = IntegrateTF(L0_pas)
# L0_pas_r = con.tf(con.balred(con.ss(L0_pas),redOrder))
# print('L0_pas_r')
# con.tf(L0_pas)

L0_pas_P = con.minreal(-TF[OutpVar+'_'+OutpVar])
L_pas_P = IntegrateTF(L0_pas_P)
# L0_pas_P_r = con.tf(con.balred(con.ss(L0_pas_P),redOrder))
print(f'L0_pas_P: ({con.dcgain(L0_pas_P):0.2f})')
con.tf(L0_pas_P)

L0_pas_Inh = con.minreal(-TF[SplitVar+'_'+SplitVar])
L_pas_Inh = IntegrateTF(L0_pas_Inh)
print(f'L0_pas_Inh: ({con.dcgain(L0_pas_Inh):0.2f})')
con.tf(L0_pas_Inh)
```

```
LL0_r (from open-loop
```

[82]:

$$\frac{5.469(s-(0.0591+0.2379j))(s-(0.0591-0.2379j))(s+2.535\times 10^{4})}{(s+(0.08215-0.06438j))(s+(0.08215+0.06438j))(s+5.04\times 10^{5})}$$

```
L0_r (from split-loop)
```

[82]:

$$\frac{5.469(s-(0.05911+0.2379j))(s-(0.05911-0.2379j))(s+2.535\times 10^{4})}{(s+(0.08214-0.06437j))(s+(0.08214+0.06437j))(s+5.04\times 10^{5})}$$

```
L0_act_r: (1.39)
```

[82]:

$$\frac{0.005713(s-(0.6279+0.494j))(s-(0.6279-0.494j))}{(s+0.08603)(s+(0.1119-0.1342j))(s+(0.1119+0.1342j))}$$

```
1 states have been removed from the model
L0_pas
L0_pas: (0.17)
```

[82]:

$$\frac{5.469(s+55.17)(s+2.539\times 10^{4})}{(s+87.72)(s+5.04\times 10^{5})}$$

```
0 states have been removed from the model
L0_pas_P: (0.17)
```

[82]:

$$\frac{5.367(s+1.627\times 10^{4})}{s+5.04\times 10^{5}}$$

```
1 states have been removed from the model
L0_pas_Inh: (-0.00)
```

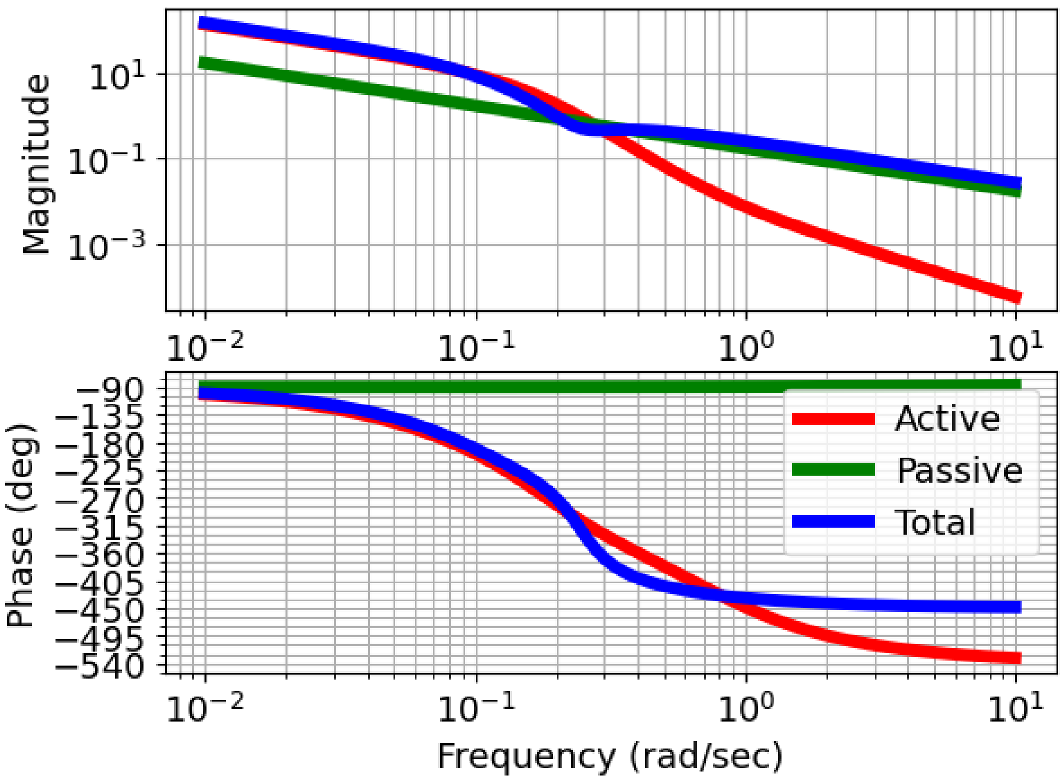
[82]:

$$\frac{0.1019(s-2.262\times 10^{-9})}{s+87.72}$$

[83]:

```python
L_list = [L_act,L_pas,LL_r]
Omega = np.logspace(-2,1,100)
mag,phase,om=con.bode_plot(L_list,omega=Omega)
plt.legend(['Active','Passive','Total','Reduced'])
# SaveFig(SystemName,'SplitBode')
```

[83]: <matplotlib.legend.Legend at 0x7253c41c2fa0>



[84]:
```
## Margins
gm, pm, wcg, wcp = con.margin(LL_r)
print(pm,wcp)
```

```
-89.798689160331 0.19851803296821272
```

[ ]:
```

```

18.5 Nichols plots

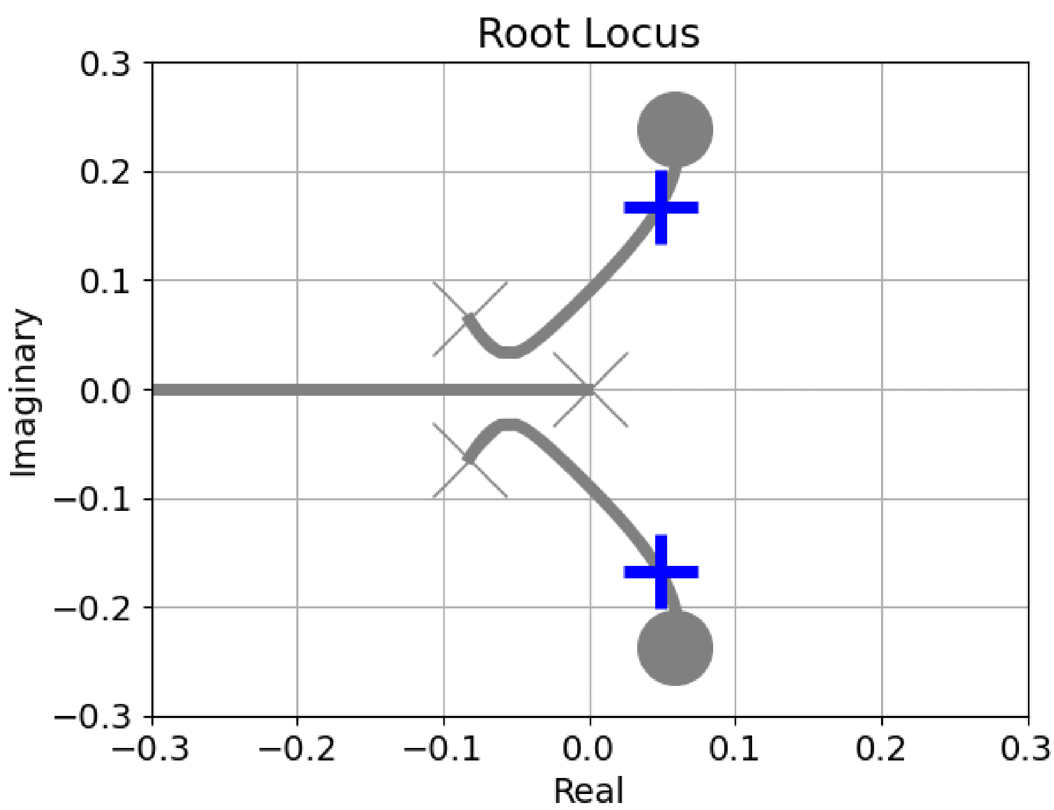
[85]:
```
# ## Nichols
# con.nichols_plot(L_list,omega)
# plt.legend(['Active','Passive','Total'])
# SaveFig(SystemName,'SplitNichols')
```

18.6 Root Locus

[86]:
```
xlim=(-0.3,0.3)
ylim=(-0.3,0.3)
kvect = np.logspace(-3,1,100)
# roots1,gains1 = con.root_locus(L_r,kvect=[1],plot=False)
# roots,gains=con.root_locus(L_r,kvect=kvect,xlim=xlim,ylim=ylim,grid=False)
# plt.plot(np.real(roots1),np.imag(roots1),color='red', marker='+',␣
↪linestyle='dashed',markersize=16)
# plt.grid()
# SetPlot(RL=True)
```

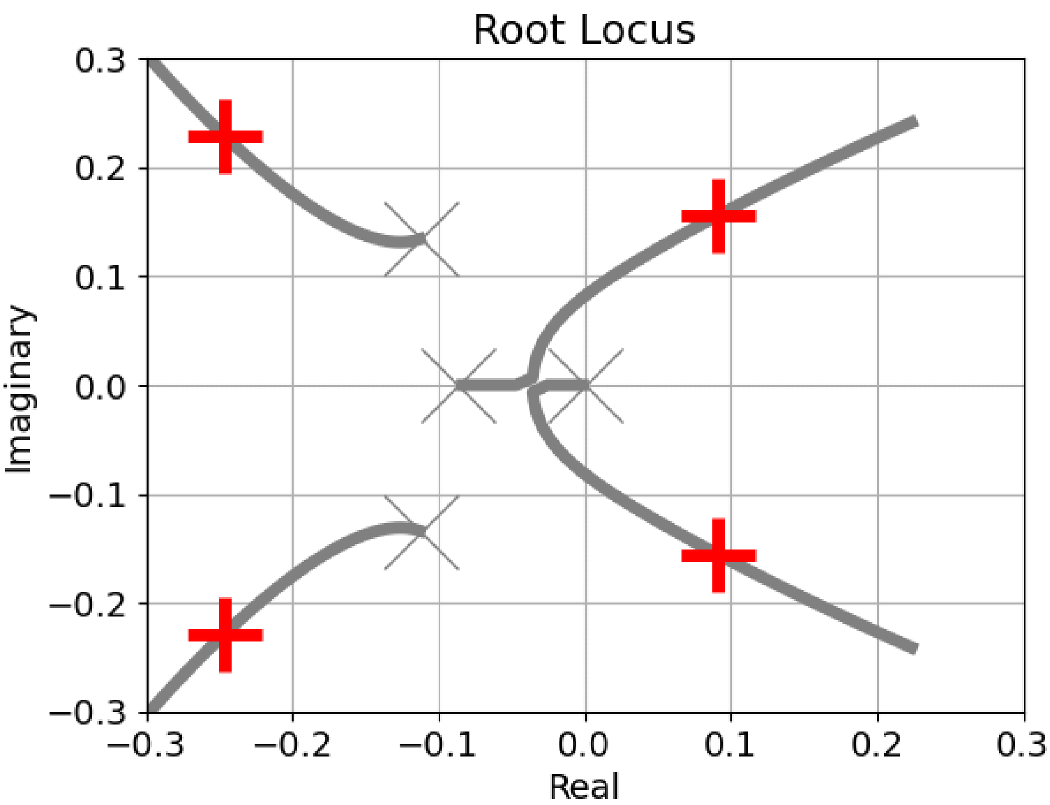
```
# SaveFig(SystemName,'SplitRootLocus',RL=True)
# SetPlot()

SetPlot(RL=True)
roots1,gains1 = con.root_locus(L,kvect=[1],plot=False)
roots,gains=con.root_locus(L_r,kvect=kvect,xlim=xlim,ylim=ylim,grid=False)
plt.plot(np.real(roots1),np.imag(roots1),color='b', marker='+',mew=5)
plt.grid()
SaveFig(SystemName,'SplitRootLocus',RL=True)
SetPlot()
```

```
[86]: [<matplotlib.lines.Line2D at 0x7253a36d4550>,
       <matplotlib.lines.Line2D at 0x7253a36d4a00>,
       <matplotlib.lines.Line2D at 0x7253a36d49d0>,
       <matplotlib.lines.Line2D at 0x7253c494e3a0>,
       <matplotlib.lines.Line2D at 0x7253c494edf0>,
       <matplotlib.lines.Line2D at 0x7253c494e610>,
       <matplotlib.lines.Line2D at 0x7253c494ef40>,
       <matplotlib.lines.Line2D at 0x7253de827160>,
       <matplotlib.lines.Line2D at 0x7253de827040>,
       <matplotlib.lines.Line2D at 0x7253de827d90>,
       <matplotlib.lines.Line2D at 0x7253de827d30>,
       <matplotlib.lines.Line2D at 0x7253de827b80>,
       <matplotlib.lines.Line2D at 0x7253de827610>,
       <matplotlib.lines.Line2D at 0x7253de8270a0>,
       <matplotlib.lines.Line2D at 0x7253c4735190>,
       <matplotlib.lines.Line2D at 0x7253c4735580>]
```



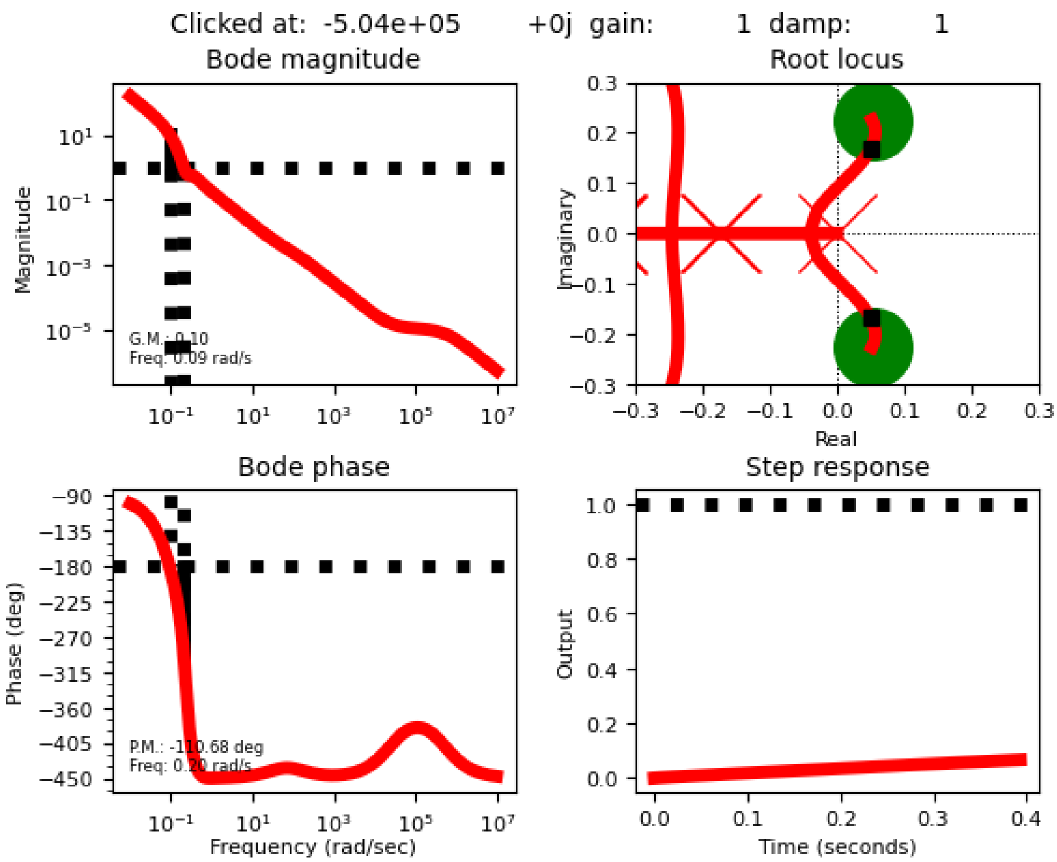
[87]:
```
xlim=(-0.3,0.3)
ylim=(-0.3,0.3)
kvect = np.logspace(-3,1,100)
# roots1,gains1 = con.root_locus(L_act,kvect=[1],plot=False)
# roots,gains=con.root_locus(L_act,kvect=kvect,xlim=xlim,ylim=ylim,grid=False)
# plt.plot(np.real(roots1),np.imag(roots1),color='red', marker='+',␣
↪linestyle='dashed',markersize=16)
# plt.grid()
# SetPlot(RL=True)
# SaveFig(SystemName,'SplitRootLocus_act',RL=True)
# SetPlot()

SetPlot(RL=True)
roots1,gains1 = con.root_locus(L_act,kvect=[1],plot=False)
roots,gains=con.root_locus(L_act,kvect=kvect,xlim=xlim,ylim=ylim,grid=False)
plt.plot(np.real(roots1),np.imag(roots1),color='r', marker='+',mew=5)
plt.grid()
SaveFig(SystemName,'SplitRootLocus_act',RL=True)
SetPlot()
```

[87]:
```
[<matplotlib.lines.Line2D at 0x7253a3b3ef70>,
 <matplotlib.lines.Line2D at 0x7253a3b3efd0>,
 <matplotlib.lines.Line2D at 0x7253a3cbf850>,
 <matplotlib.lines.Line2D at 0x7253a3cbfa90>]
```



18.7 Sisotool

[88]:
```
con.sisotool(L,xlim_rlocus=xlim,ylim_rlocus=ylim)
```



18.8 Bode (replotted)

[89]:
```
Name = ['Active','Passive','Total','Reduced']
# Omega = np.logspace(-2,0)
for i,l in enumerate(L_list):
#     print(i,l)
    mag,phase,omega = con.bode_plot(l,omega=Omega,plot=False)
    plt.loglog(omega,mag,label=Name[i])
# plt.hlines(1,min(omega),max(omega),ls='dashed',color='black',label='Unit␣
 ↪gain')
plt.hlines(1,wcp/10,wcp*10,ls='dashed',color='black',lw=2)
plt.vlines(wcp,0.1,10,ls='dashed',color='black',lw=2)
plt.legend(loc='lower left')
# plt.legend()
plt.grid()
# plt.xlabel(r'$\omega$')
```

```
# print(f'{wcp:.2f}')
# print(f'$\\omega$ rad/sec ($\\omega_c = {int(wcp)}$)')
plt.xlabel(f'$\\omega$ rad/sec ($\\omega_c = {wcp:.2f}$)')
plt.ylabel(r'$|L|$')
SaveFig(SystemName,'SplitBodeMag')
```

[89]: [<matplotlib.lines.Line2D at 0x7253a2753a60>]

[89]: [<matplotlib.lines.Line2D at 0x7253a2753df0>]

[89]: [<matplotlib.lines.Line2D at 0x7253a2768100>]

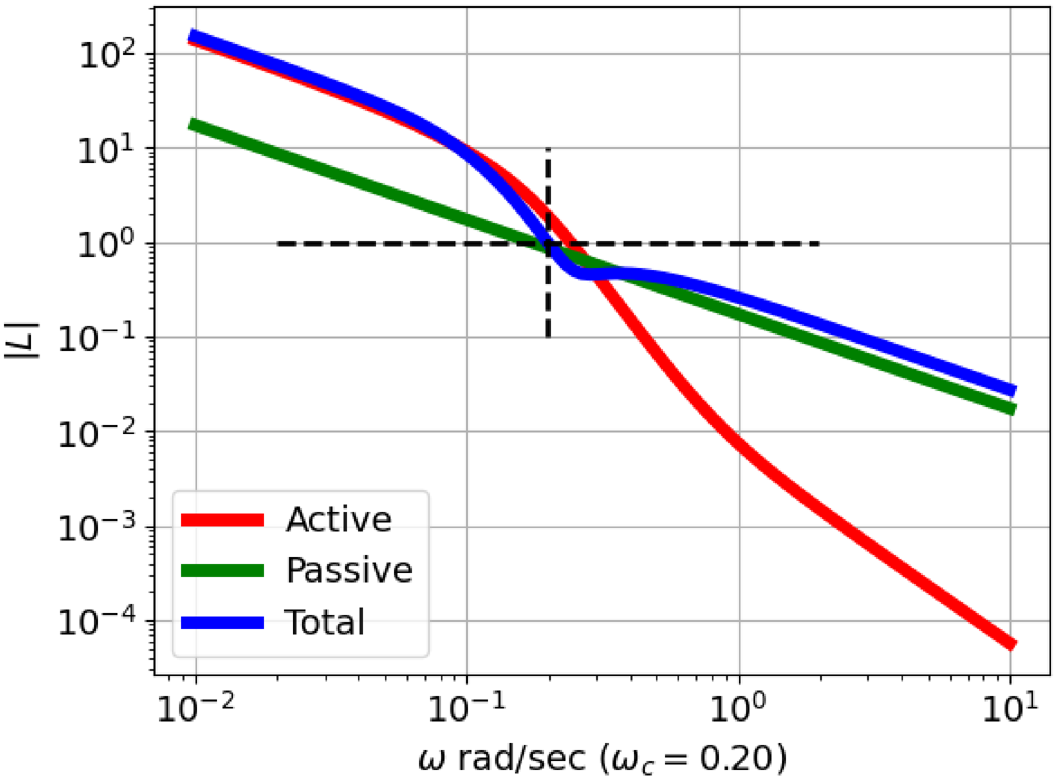
[89]: <matplotlib.collections.LineCollection at 0x7253a27b4460>

[89]: <matplotlib.collections.LineCollection at 0x7253a2753fd0>

[89]: <matplotlib.legend.Legend at 0x7253a2946be0>

[89]: Text(0.5, 0, '$\\omega$ rad/sec ($\\omega_c = 0.20$)')

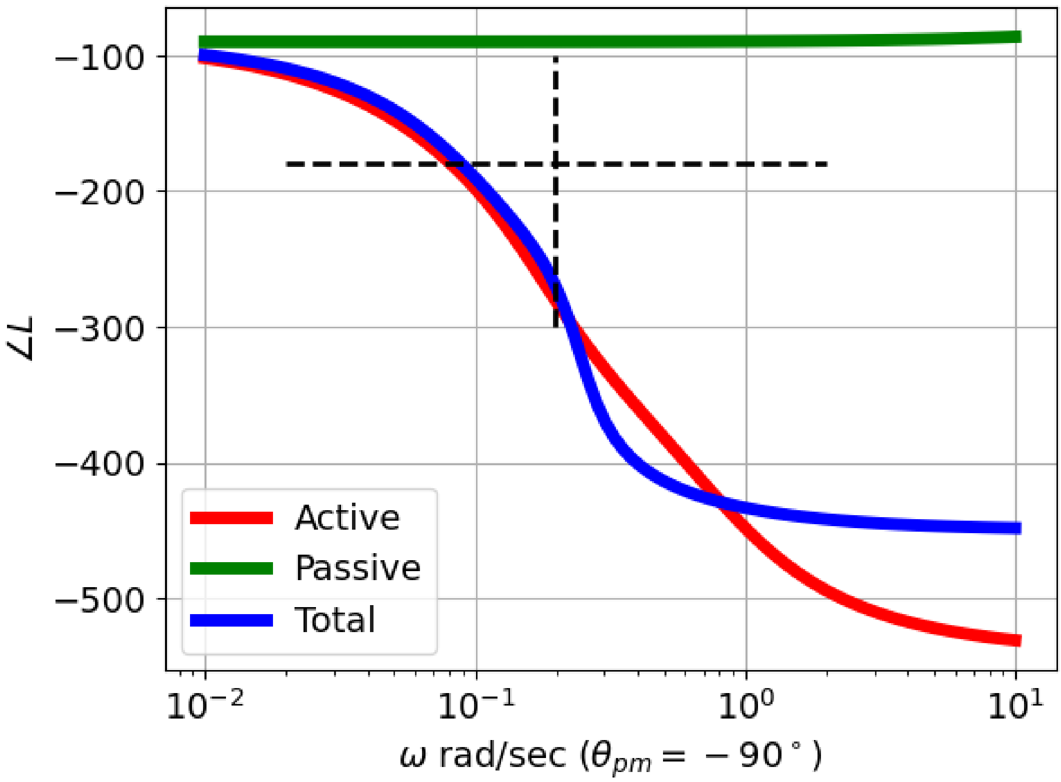
[89]: Text(0, 0.5, '$|L|$')



[90]:
```
for i,l in enumerate(L_list):
#     print(i,l)
    mag,phase,omega= con.bode_plot(l,omega=Omega,plot=False)
    plt.semilogx(omega,phase*180/np.pi,label=Name[i])
```

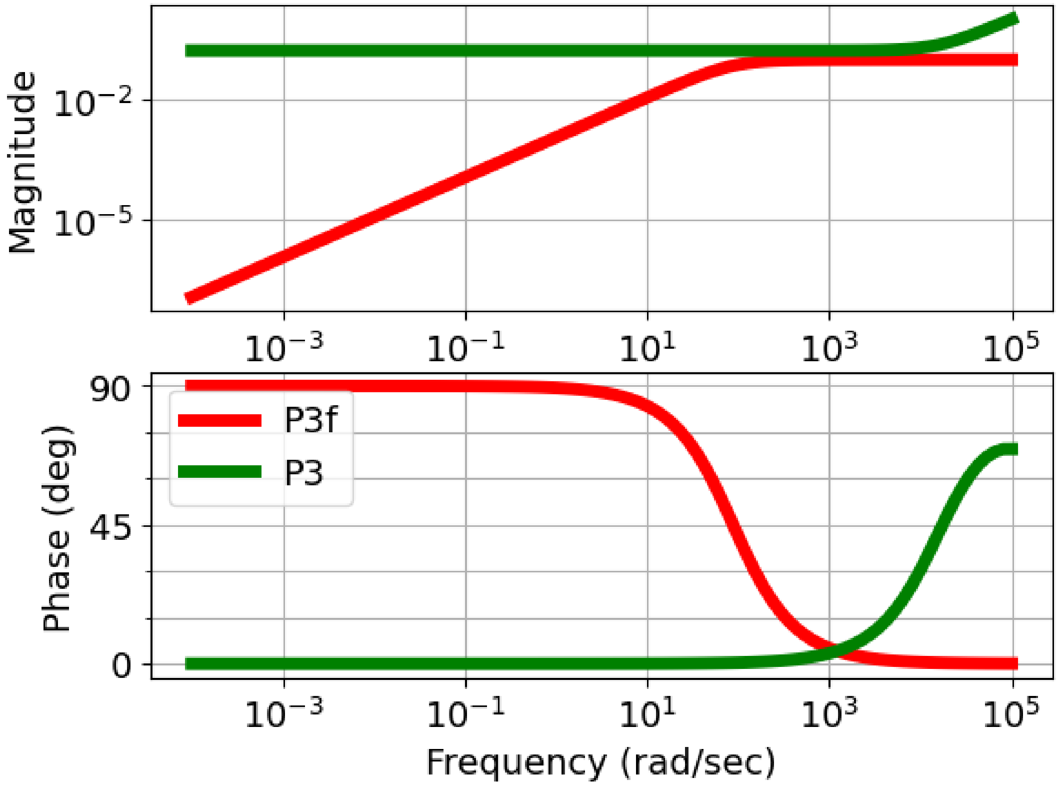
```
plt.legend()
# plt.hlines(-180,min(omega),max(omega),ls='dashed',color='black')
plt.hlines(-180,wcp/10,wcp*10,ls='dashed',color='black',lw=2)
plt.vlines(wcp,-300,-100,ls='dashed',color='black',lw=2)
# plt.xlabel(r'$\omega$')
print(f'{pm:.0f}')
plt.xlabel(f'$\\omega$ rad/sec ($\\theta_{{pm}} = {pm:.0f}^\circ$)')
plt.ylabel(r'$\angle{L}$')
plt.grid()
SaveFig(SystemName,'SplitBodePha')
```

```
[90]: [<matplotlib.lines.Line2D at 0x7253a249fa60>]

[90]: [<matplotlib.lines.Line2D at 0x7253a249f940>]

[90]: [<matplotlib.lines.Line2D at 0x7253a249ff70>]

[90]: <matplotlib.legend.Legend at 0x7253a2565b80>

[90]: <matplotlib.collections.LineCollection at 0x7253a249ffa0>

[90]: <matplotlib.collections.LineCollection at 0x7253a24b20a0>

-90

[90]: Text(0.5, 0, '$\\omega$ rad/sec ($\\theta_{pm} = -90^\\circ$)')

[90]: Text(0, 0.5, '$\\angle{L}$')
```



[91]:
```
L_list = [L0_pas_Inh,L0_pas_P]
Omega = np.logspace(-4,5,100)
mag,phase,om=con.bode_plot(L_list,omega=Omega,initial_phase=90)
plt.legend(['P3f','P3'])
# SaveFig(SystemName,'SplitBode')
```

[91]: <matplotlib.legend.Legend at 0x7253a236bc40>

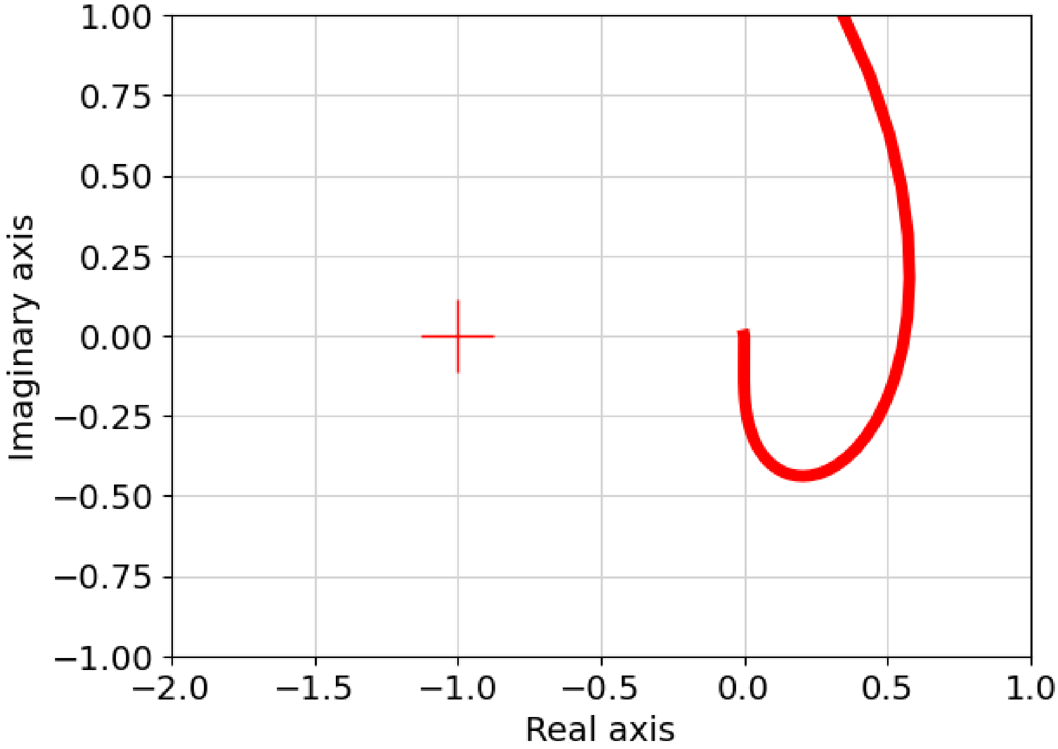


[92]:
```
con.nyquist_plot([LL],mirror_style=False)
plt.xlim(-2,1)
plt.ylim(-1,1)
```

[92]: 2

[92]: (-2.0, 1.0)

[92]: (-1.0, 1.0)

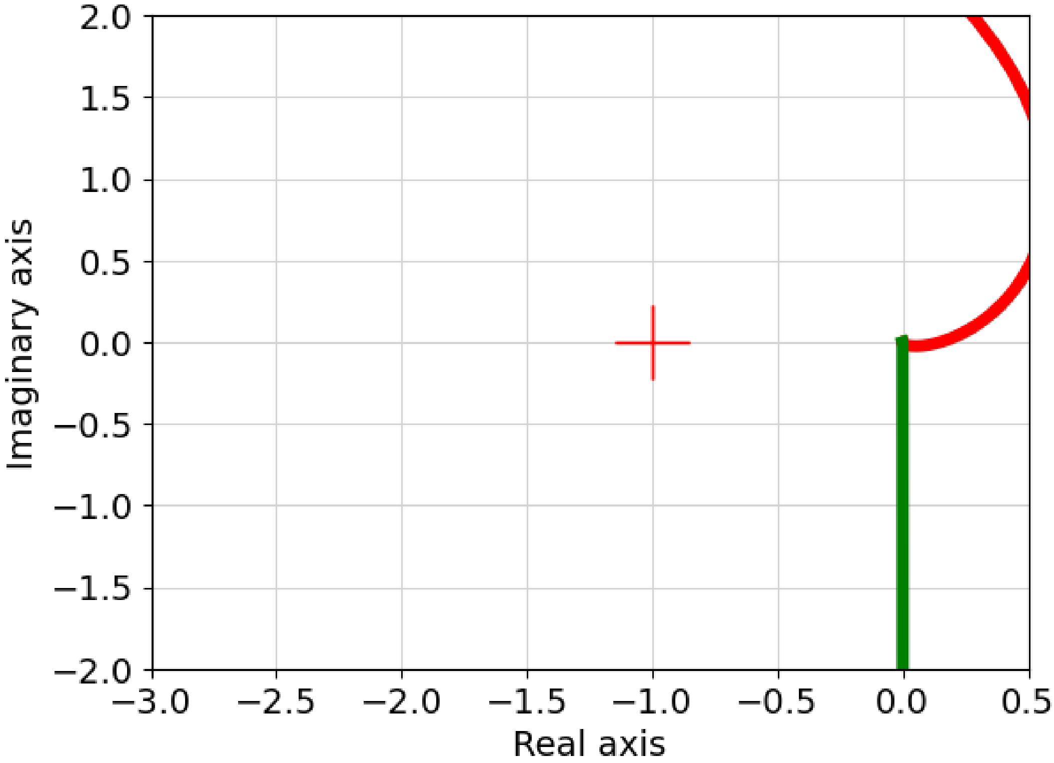


[93]:
```
con.nyquist_plot([L_act,L_pas],mirror_style=False)
plt.xlim(-3,0.5)
plt.ylim(-2,2)
```

[93]: [2, 0]

[93]: (-3.0, 0.5)

[93]: (-2.0, 2.0)



[94]:
```
con.tf(L0_r)
con.tf(con.balred(L0_r,2))
```

[94]:

$$\frac{5.469(s-(0.05911+0.2379j))(s-(0.05911-0.2379j))(s+2.535\times 10^{4})}{(s+(0.08214-0.06437j))(s+(0.08214+0.06437j))(s+5.04\times 10^{5})}$$

[94]:

$$\frac{5.469(s+0.1267)(s+2.535\times 10^{4})}{(s+0.01552)(s+5.04\times 10^{5})}$$

References