Simulation Open Loop

In [276…
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from package_DBR import FOPDT, SOPDT, Process, Bode
```

Load Data From Experiment

MV

In [279…
```
nameFile = 'Cleaned_data_Open_loop_experiment_on_MV_2025-03-10-14h08.txt'

titleName = nameFile.split('.')[0]
data = pd.read_csv('Data/' + nameFile)

if 'MV' in nameFile:
    ExpVariable = 'MV'
    tm = data['tm'].values
    MVm = data['MVm'].values
    PVm = data['PVm'].values
else:
    ExpVariable = 'DV'
    tm = data['tm'].values
    DVm = data['DVm'].values
    PVm = data['PVm'].values

print(ExpVariable)
```

```
MV
```

DV

In [281…
```
nameFile = 'Cleaned_data_Open_loop_experiment_on_DV_2025-03-10-16h13.txt'

titleName = nameFile.split('.')[0]
data = pd.read_csv('Data/' + nameFile)

if 'MV' in nameFile:
    ExpVariable = 'MV'
    tm = data['tm'].values
    MVm = data['MVm'].values
    PVm = data['PVm'].values
else:
    ExpVariable = 'DV'
    tm_DV = data['tm'].values
    DVm = data['DVm'].values
    PVm_DV = data['PVm'].values

print(ExpVariable)
```

```
DV
```

Optimal Parameters

In [283…
```
Ts = 1
```

MV

In [285…
```
#MV

Kp_OMV_FOPDT = 0.5280686804191848
T_OMV_FOPDT = 160.98841281582534
theta_OMV_FOPDT = 35.21872295460488

Kp_OMV_SOPDT = 0.5279783321046589
T1_OMV_SOPDT = 156.72135296437907
T2_OMV_SOPDT = 22.414015717250276
theta_OMV_SOPDT = 15.732227706547503

#Process values
PV_OMV_FOPDT = FOPDT(MVm, Kp_OMV_FOPDT, T_OMV_FOPDT, theta_OMV_FOPDT, Ts)
PV_OMV_SOPDT = SOPDT(MVm, Kp_OMV_SOPDT, T1_OMV_SOPDT, T2_OMV_SOPDT, theta_OMV_SOPDT, Ts)
```

DV

In [287…
```
#DV

Kp_ODV_FOPDT = 0.6153119961469684
T_ODV_FOPDT = 163.91357248141884
theta_ODV_FOPDT = 32.38845620532809

Kp_ODV_SOPDT =  0.611563683291308
T1_ODV_SOPDT = 155.0912717619001
T2_ODV_SOPDT = 25.53958809347822
theta_ODV_SOPDT = 11.559549786267498

#Process values
PV_ODV_FOPDT = FOPDT(DVm, Kp_ODV_FOPDT, T_ODV_FOPDT, theta_ODV_FOPDT, Ts)
PV_ODV_SOPDT = SOPDT(DVm, Kp_ODV_SOPDT, T1_ODV_SOPDT, T2_ODV_SOPDT, theta_ODV_SOPDT, Ts)
```

Graphic's Parameters - MV

In [289…
```
t1 = 95
t2 = 133
Tu = 17
Tg = 228 - Tu
a = 0.1

## Temporary
Kp = 0.5101
```

Broida 1

In [291…
```
T_B1 = Tg
theta_B1 = Tu
```

```
print(f'T = {T_B1}, theta = {theta_B1}')

#Process values
PV_B1 = FOPDT(MVm, Kp, T_B1, theta_B1, Ts)
```

```
T = 211, theta = 17
```

Broida 2

In [293…

```
T_B2 = 5.5*(t2-t1)
theta_B2 = 2.8*t1 - 1.8*t2

print(f'T = {T_B2}, theta = {theta_B2}')

#Process values
PV_B2 = FOPDT(MVm, Kp, T_B2, theta_B2, Ts)
```

```
T = 209.0, theta = 26.599999999999994
```

van der Grinten

In [295…

```
T1_G = Tg * (3 * a * np.exp(1) - 1)/(1 + a * np.exp(1))
T2_G = Tg*(1-a*np.exp(1))/(1+a*np.exp(1))
theta_G = Tu - (T1_G*T2_G)/(T1_G + 3*T2_G)

print(f'T1 = {T1_G}, T2 = {T2_G}, theta = {theta_G}')

#Process values
PV_G = SOPDT(MVm, Kp, T1_G, T2_G, theta_G, Ts)
```

```
T1 = -30.611650837477573, T2 = 120.80582541873879, theta = 28.145270708206365
```

Strejc

In [297…

```
ratio = Tu/Tg

#Computation of Strejc model parameters
table_a = {1 : 0, 2 : 0.1, 3 : 0.22, 4 : 0.32, 5 : 0.41, 6 : 0.49, 7 : 0.57}
table_b = {1 : 1, 2 : 2.72, 3 : 3.69, 4 : 4.46, 5 : 5.12, 6 : 5.7, 7 : 6.23}

#Order
n = 0

for key, value in table_a.items() :
    if value <= ratio and ratio < table_a[key+1] :
        n = key

a_n = table_a[n]
b_n = table_b[n]

T_S = Tg/b_n
Tuth= a_n*Tg
theta_S = Tu - Tuth

print(f'order = {n}, T = {T_S}, theta = {theta_S}')

#Process values
PV_S = FOPDT(MVm, Kp, T_S, theta_S, Ts)
```

```
order = 1, T = 211.0, theta = 17
```

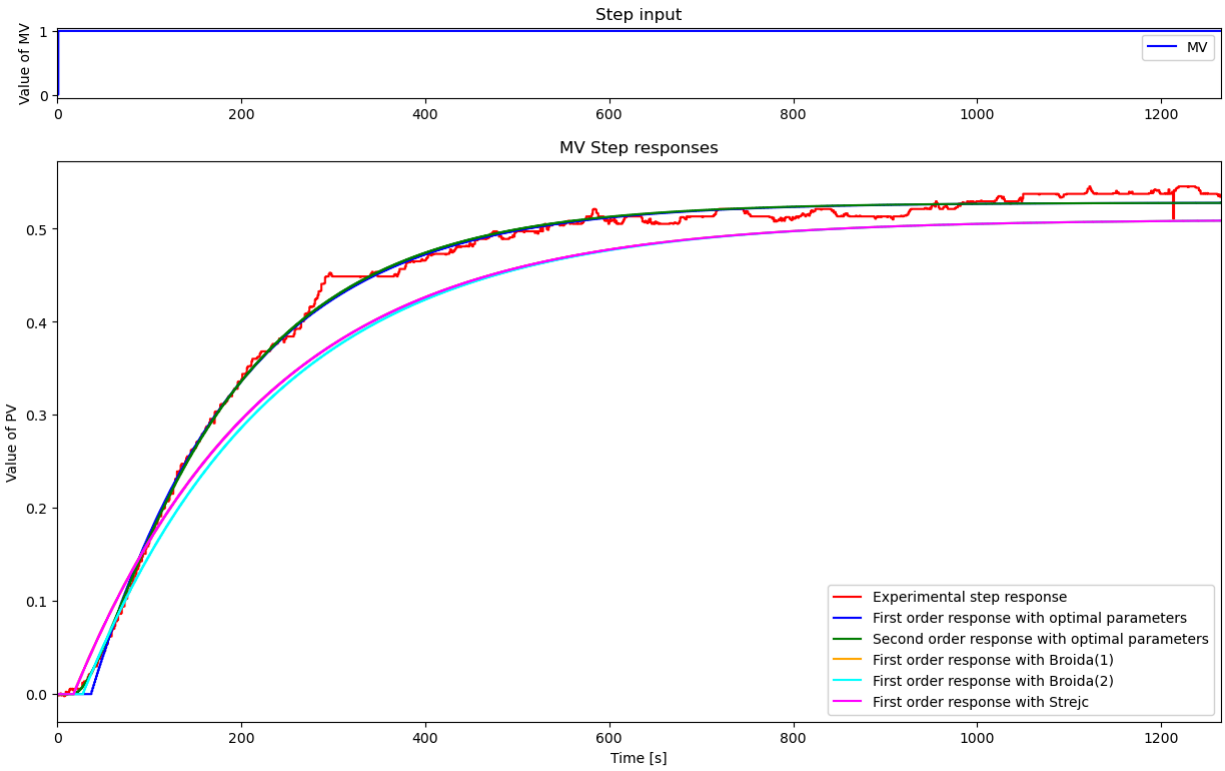
Manipulated Value

In [299…

```
fig, axs = plt.subplots(2, 1, figsize=(15, 9), gridspec_kw={'height_ratios': [1, 8]})

axs[0].step(tm, MVm, 'b-', label='MV', where='post')
axs[0].set_ylabel('Value of MV')
axs[0].set_title('Step input')
axs[0].legend(loc='best')
axs[0].set_xlim([0, np.max(tm)])

axs[1].step(tm, PVm, 'Red', label='Experimental step response', where='post')
axs[1].step(tm, PV_OMV_FOPDT, 'Blue', label='First order response with optimal parameters', where='post')
axs[1].step(tm, PV_OMV_SOPDT, 'Green', label='Second order response with optimal parameters', where='post')
axs[1].step(tm, PV_B1, 'Orange', label='First order response with Broida(1)', where='post')
axs[1].step(tm, PV_B2, 'Cyan', label='First order response with Broida(2)', where='post')
#axs[1].step(tm, PV_G, 'r-', label='Second order response with van der Grinten', where='post')
axs[1].step(tm, PV_S, 'Magenta', label='First order response with Strejc', where='post')
axs[1].set_ylabel('Value of PV')
axs[1].set_xlabel('Time [s]')
axs[1].set_title('MV Step responses')
axs[1].legend(loc='best')
axs[1].set_xlim([0, np.max(tm)])
```

Out[299…

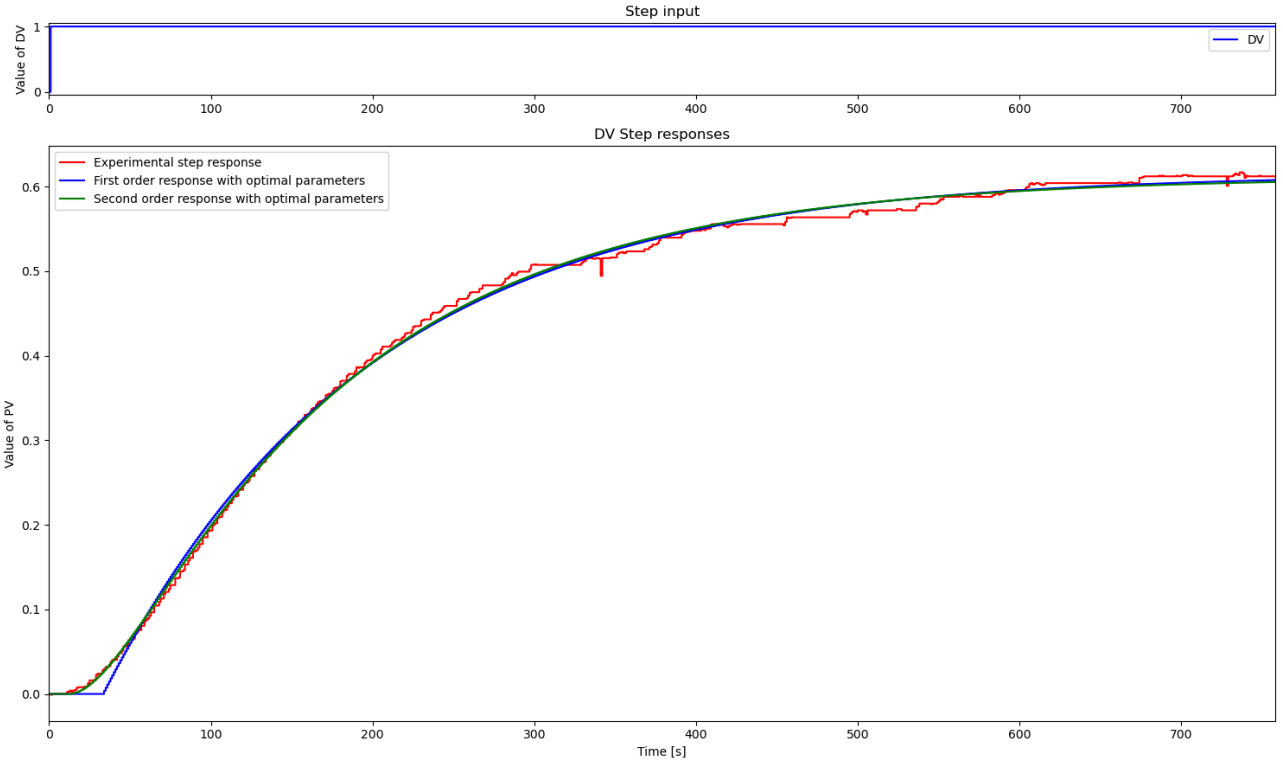
```
(0.0, 1265.0)
```



Disturbance Value

In [301…

```
fig, axs = plt.subplots(2, 1, figsize=(15, 9), gridspec_kw={'height_ratios': [1, 8]})

axs[0].step(tm_DV, DVm, 'b-', label='DV', where='post')
axs[0].set_ylabel('Value of DV')
axs[0].set_title('Step input')
axs[0].legend(loc='best')
axs[0].set_xlim([0, np.max(tm_DV)])

axs[1].step(tm_DV, PVm_DV, 'Red', label='Experimental step response', where='post')
axs[1].step(tm_DV, PV_ODV_FOPDT, 'Blue', label='First order response with optimal parameters', where='post')
axs[1].step(tm_DV, PV_ODV_SOPDT, 'Green', label='Second order response with optimal parameters', where='post')
axs[1].set_ylabel('Value of PV')
axs[1].set_xlabel('Time [s]')
axs[1].set_title('DV Step responses')
axs[1].legend(loc='best')
axs[1].set_xlim([0, np.max(tm_DV)])

fig.tight_layout()
```



Bode graphs

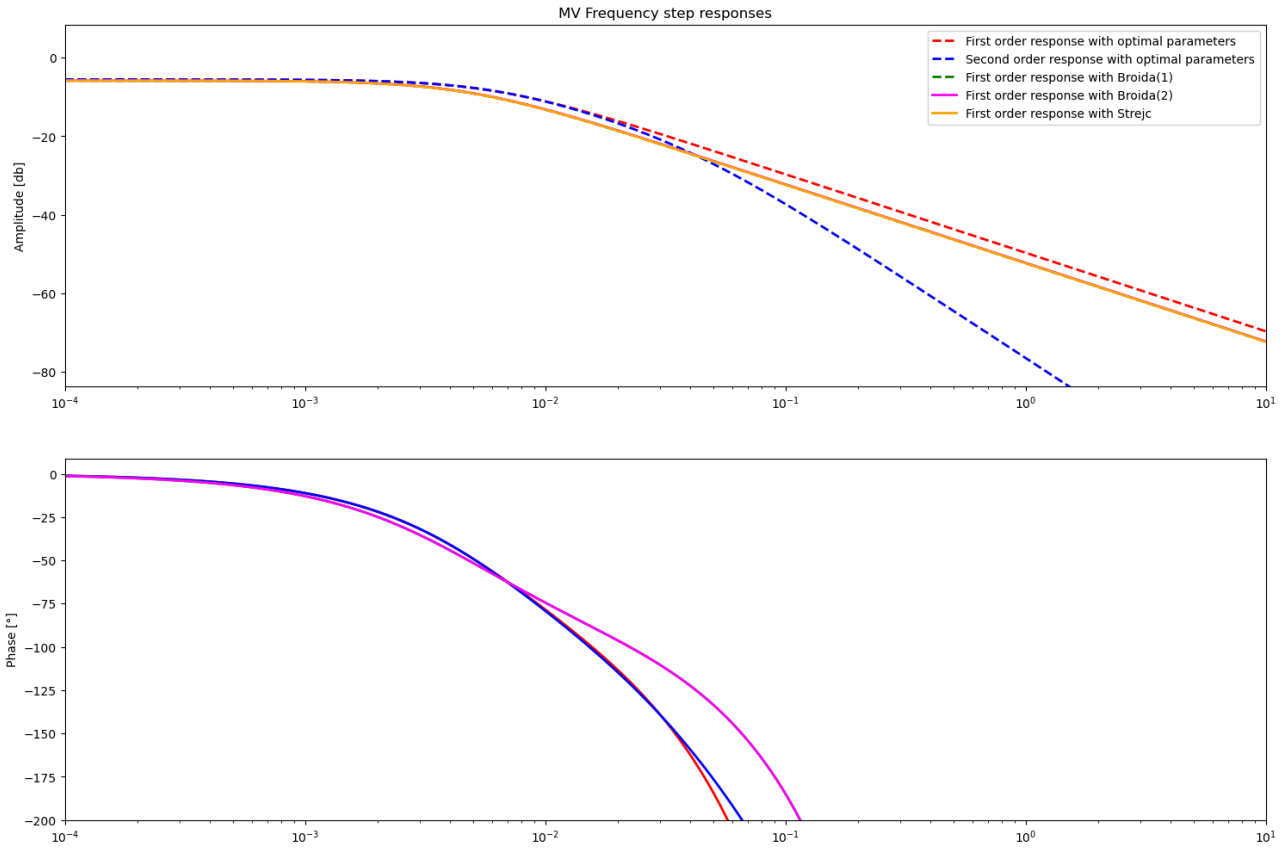
Pour MV

In [304…

```
P_OMV_FOPDT = Process({'Kp' : Kp_OMV_FOPDT, 'theta' : theta_OMV_FOPDT, 'Tlag1' : T_OMV_FOPDT})
P_OMV_SOPDT = Process({'Kp' : Kp_OMV_SOPDT, 'theta' : theta_OMV_SOPDT, 'Tlag1' : T1_OMV_SOPDT, 'Tlag2' : T2_OMV_SOPDT})
P_B1 = Process({'Kp' : Kp, 'theta' : theta_B1, 'Tlag1' : T_B1})
P_B2 = Process({'Kp' : Kp, 'theta' : theta_B2, 'Tlag1' : T_B2})
P_S = Process({'Kp' : Kp, 'theta' : theta_S, 'Tlag1' : T_S})

#omega = np.logspace(-7, 20, 15000)
#omega = np.logspace(-4, 1, 10000)
omega = np.logspace(-4, 1, 10000)

Ps_OMV_FOPDT = Bode(P_OMV_FOPDT, omega, False)
Ps_OMV_SOPDT = Bode(P_OMV_SOPDT, omega, False)
Ps_B1 = Bode(P_B1, omega, False)
Ps_B2 = Bode(P_B2, omega, False)
Ps_S = Bode(P_S, omega, False)

plt.figure(figsize = (18,12))

# Gain  plot
plt.subplot(2,1,1)
gain_min = np.min(20*np.log10(np.abs(Ps_OMV_FOPDT)/5))
gain_max = np.max(20*np.log10(np.abs(Ps_OMV_FOPDT)*5))
plt.semilogx(omega,20*np.log10(np.abs(Ps_OMV_FOPDT)),'r--',linewidth=2,label='First order response with optimal parameters')
plt.semilogx(omega,20*np.log10(np.abs(Ps_OMV_SOPDT)),'b--',linewidth=2,label='Second order response with optimal parameters')
plt.semilogx(omega,20*np.log10(np.abs(Ps_B1)),'g--',linewidth=2,label='First order response with Broida(1)')
plt.semilogx(omega,20*np.log10(np.abs(Ps_B2)),'Magenta',linewidth=2,label='First order response with Broida(2)')
plt.semilogx(omega,20*np.log10(np.abs(Ps_S)),'Orange',linewidth=2,label='First order response with Strejc')
plt.xlim([np.min(omega), np.max(omega)])
plt.ylim([gain_min, gain_max])
plt.ylabel('Amplitude [db]')
plt.title('MV Frequency step responses')
plt.legend(loc='best')

# Phase plot
plt.subplot(2,1,2)
ph_min = np.min((180/np.pi)*np.unwrap(np.angle(Ps_OMV_FOPDT))) - 10
ph_max = np.max((180/np.pi)*np.unwrap(np.angle(Ps_OMV_FOPDT))) + 10
plt.semilogx(omega, (180/np.pi)*np.unwrap(np.angle(Ps_OMV_FOPDT)),'Red',linewidth=2)
plt.semilogx(omega, (180/np.pi)*np.unwrap(np.angle(Ps_OMV_SOPDT)),'Blue',linewidth=2)
plt.semilogx(omega, (180/np.pi)*np.unwrap(np.angle(Ps_B1)),'Green',linewidth=2)
plt.semilogx(omega, (180/np.pi)*np.unwrap(np.angle(Ps_S)),'Magenta',linewidth=2)
plt.xlim([np.min(omega), np.max(omega)])
plt.ylim([np.max([ph_min, -200]), ph_max])
plt.ylabel('Phase [°]')
```

Out[304…

```
Text(0, 0.5, 'Phase [°]')
```



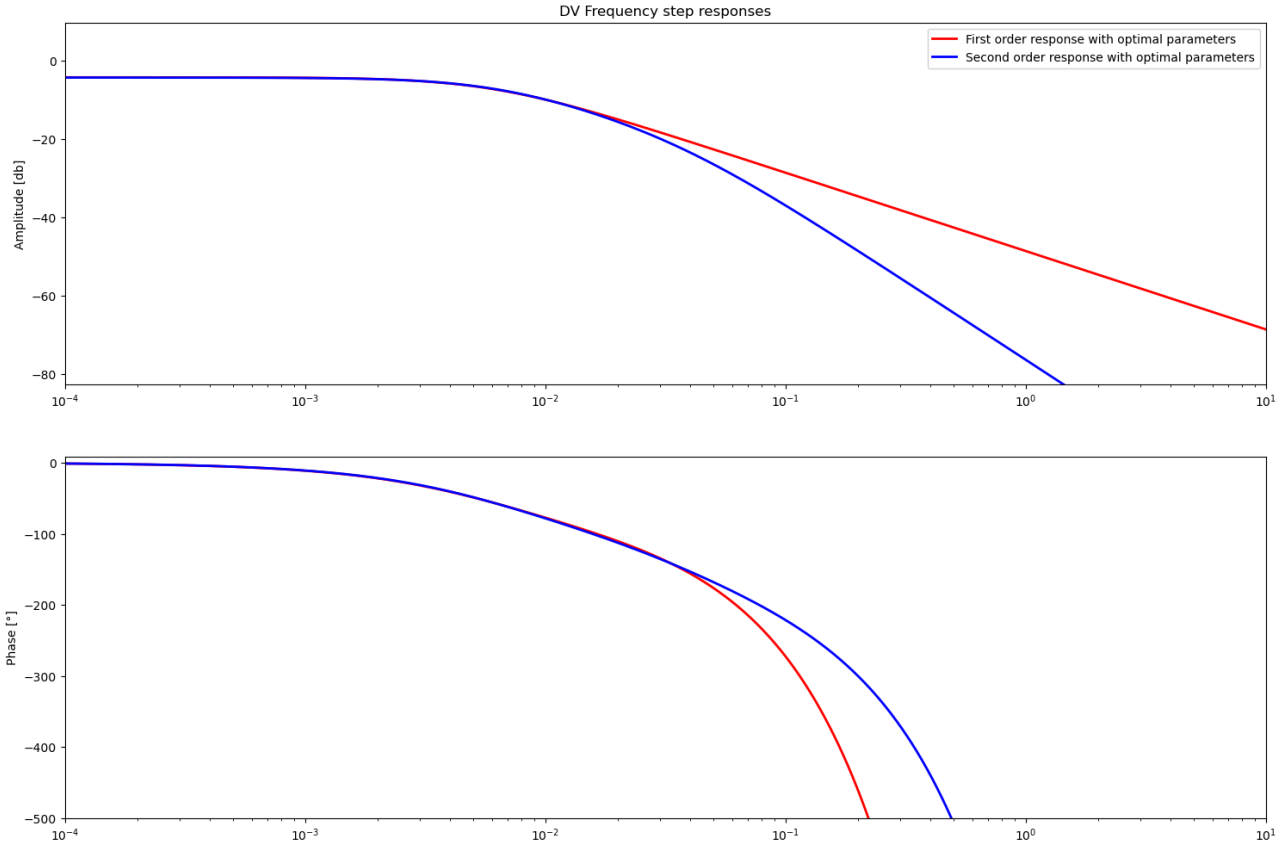
Pour DV

In [306…

```
P_ODV_FOPDT = Process({'Kp' : Kp_ODV_FOPDT, 'theta' : theta_ODV_FOPDT, 'Tlag1' : T_ODV_FOPDT})
P_ODV_SOPDT = Process({'Kp' : Kp_ODV_SOPDT, 'theta' : theta_ODV_SOPDT, 'Tlag1' : T1_ODV_SOPDT, 'Tlag2' : T2_ODV_SOPDT})

#omega = np.logspace(-7, 5, 10000)
omega = np.logspace(-4, 1, 10000)
Ps_ODV_FOPDT = Bode(P_ODV_FOPDT, omega, False) # Optimal Broida
Ps_ODV_SOPDT = Bode(P_ODV_SOPDT, omega, False) # Optimal van der Grinten

plt.figure(figsize = (18,12))

# Gain plot
plt.subplot(2,1,1)
gain_min = np.min(20*np.log10(np.abs(Ps_ODV_FOPDT)/5))
gain_max = np.max(20*np.log10(np.abs(Ps_ODV_FOPDT)*5))
plt.semilogx(omega,20*np.log10(np.abs(Ps_ODV_FOPDT)),'Red',linewidth=2,label='First order response with optimal parameters')
plt.semilogx(omega,20*np.log10(np.abs(Ps_ODV_SOPDT)),'Blue',linewidth=2,label='Second order response with optimal parameters')
plt.xlim([np.min(omega), np.max(omega)])
plt.ylim([gain_min, gain_max])
plt.ylabel('Amplitude [db]')
```

```
plt.title('DV Frequency step responses')
plt.legend(loc='best')

# Phase plot
plt.subplot(2,1,2)
ph_min = np.min((180/np.pi)*np.unwrap(np.angle(Ps_ODV_FOPDT))) - 10
ph_max = np.max((180/np.pi)*np.unwrap(np.angle(Ps_ODV_FOPDT))) + 10
plt.semilogx(omega, (180/np.pi)*np.unwrap(np.angle(Ps_ODV_FOPDT)),'Red',linewidth=2)
plt.semilogx(omega, (180/np.pi)*np.unwrap(np.angle(Ps_ODV_SOPDT)),'Blue',linewidth=2)
plt.xlim([np.min(omega), np.max(omega)])
plt.ylim([np.max([ph_min, -500]), ph_max])
plt.ylabel('Phase [°]')
```

Out[306… `Text(0, 0.5, 'Phase [°]')`



In [ ]: