vreg

December 6, 2023

0.1 Background and Motivation

The purpose of autonomously adjusting reference voltage (AARV) is to mitigate rapid voltage change (RVC), without affecting the utility's control of steady-state voltage or the DER's steady-state reactive power. AARV may be used with a volt-watt function that mitigates steady-state overvoltage, but otherwise AARV does not regulate steady-state voltage.

Page 39 of IEEE 1547-2018 says: *The DER shall be capable of autonomously adjusting reference voltage (VRef) with VRef being equal to the low pass filtered measured voltage. The time constant shall be adjustable at least over the range of 300 s to 5000 s. The voltage-reactive power Volt-Var curve characteristic shall be adjusted autonomously as VRef changes. The approval of the Area EPS operator shall be required for the DER to autonomously adjust the reference voltage. Implementation of the autonomous VRef adjustability and the associated time constant shall be specified by the Area EPS operator.*

References:

- 2019 Conference Paper: https://doi.org/10.1109/PVSC40753.2019.8981277
- 2023 Conference Paper: https://doi.org/10.1109/PESGM52003.2023.10252317

Run the following cell to define support functions using Numpy and Matplotlib

[14]:
```
import sys
import os
import matplotlib.pyplot as plt
import numpy as np
import math

# convert center, deadband, slope, and q limits into a table of V and Q points.
# the function returns two arrays for the V and Q points
#    the arrays have sentinel elements below V1 and above V4, so they are 6␣
 ↪elements long (not 4)
#    the sentinel elements clarify that constant extrapolation is used outside␣
 ↪the range [V1..V4]
def set_characteristic (center=1.0, deadband=0.0, slope=22.0, qmax=0.44,␣
 ↪qmin=-0.44, qbias=0.0):
  if qbias > qmax:
    qbias = qmax
  elif qbias < qmin:
    qbias = qmin
```

```python
    Q1 = qmax
    Q2 = qbias
    Q3 = qbias
    Q4 = qmin
    V2 = center - 0.5 * deadband
    V3 = center + 0.5 * deadband
    V1 = V2 - (Q1 - Q2) / slope
    V4 = V3 - (Q4 - Q3) / slope
    VL = V1 - 0.01
    VH = V4 + 0.01
    vtable = np.array ([VL, V1, V2, V3, V4, VH])
    qtable = np.array ([Q1, Q1, Q2, Q3, Q4, Q4])
    return vtable, qtable

# This function plots a volt-var characteristic, and returns the VQ points.
# (Tabular display is not needed in this notebook.)
def show_characteristic (label, center, deadband, slope, qmax, qmin, qbias=0):
    vtable, qtable = set_characteristic (center, deadband, slope, qmax, qmin,␣
 ↪qbias)

    # bounds for plotting the horizontal axis
    vmin = vtable[0]-0.01
    vmax = vtable[-1]+0.01

    # evaluate the characteristic over 500 equal voltage intervals
    v = np.linspace (vmin, vmax, 501)
    # interpolating Q using the numpy library function
    q = np.interp (v, vtable, qtable)

    # create the plot
    fig, ax = plt.subplots(1, 1, figsize=(6,4), tight_layout=True)
    fig.suptitle ('{:s} volt-var characteristic'.format (label))
    ax.plot (vtable, qtable, marker='o', color='blue', label='Points and␣
 ↪Sentinels')
    ax.plot (v, q, color='red', label='Interpolated')
    ax.grid ()
    ax.set_xlabel ('V [pu]')
    ax.set_ylabel ('Q [pu]')
    ax.set_xlim (vmin, vmax)
    ax.legend ()

    plt.show ()
    return vtable, qtable

# This function calculates V and dV at the POC from the formula
# in clause 5.1.2 of IEEE 1547.2-2023
# All input and output in per-unit
```

```
def get_voltage (rpu, xpu, dP, dQ, vsrcpu):
  a1 = vsrcpu*vsrcpu + rpu*dP + xpu*dQ
  a2 = xpu*dP - rpu*dQ
  d = math.sqrt (a1*a1 + a2*a2) / vsrcpu / vsrcpu - 1.0
  vpu = vsrcpu + d
  return vpu, d

# use the current directory as default location for the "save plot" buttons
# optimize the graphic export for LaTex and PDF
plt.rcParams['savefig.directory'] = os.getcwd()
plt.rcParams['savefig.pad_inches'] = 0.05
plt.rcParams['savefig.dpi'] = 300.0
plt.rcParams['savefig.bbox'] = 'tight'
# invoke the Jupyter support for  Matplotlib graphics
%matplotlib widget
```

0.2 Global Variables

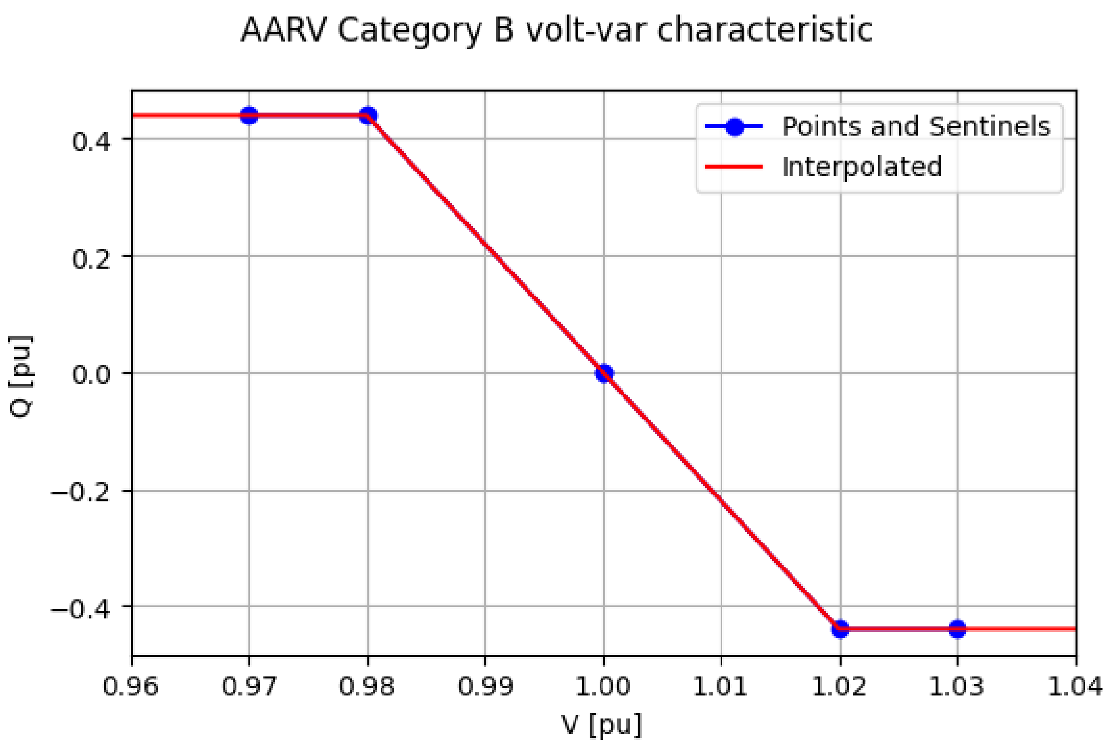
Run the following cell to define and test variables for:

- A category B volt-var characteristic with maximum allowed slope and zero deadband.
- A 6-MW battery energy storage system (BESS) for testing the AARV response to system conditions.

[15]:
```
# GLOBAL variables for the Nantucket BESS example; only the volt-var parameters␣
↪will change in the notebook examples
SBASE = 6.0e6
VBASE = 13200.0
ZBASE = VBASE*VBASE/SBASE
RPU = 1.210 / ZBASE # 1.744
XPU = 2.8339 / ZBASE # 2.6797
DP = 6.0e6 / SBASE
DQ = 0.0 / SBASE
VSRC = 1.0

vpoc, d = get_voltage (RPU, XPU, DP, DQ, VSRC)

print ('Nantucket BESS example for dP={:.3f}, dQ={:.3f}, R={:.3f}, X={:.3f} pu:␣
↪Vpoc={:.4f}, dV={:.4f} pu'.format (DP, DQ, RPU, XPU, vpoc, d))
# pseudo-code from OpenDSS
#Verr = Vpu - Vreg
#Vreg = Vreg + Verr * IncRef
#Qtarg = -K * (V - Vreg) + Qbias # differs from OpenDSS in that it does not␣
↪iterate
#Q = Qlast + (Q - Qlast) * IncOL

# plot the default characteristic used for AARV examples
```

```
vtable, qtable = show_characteristic ('AARV Category B', center=1, deadband=0,
    slope=22, qmax=0.44, qmin=-0.44)
```

```
Nantucket BESS example for dP=1.000, dQ=0.000, R=0.042, X=0.098 pu: Vpoc=1.0462,
dV=0.0462 pu
```



0.3 AARV Test from 1547.1-2020

Clause 5.14.5 of IEEE 1547.1-2020 describes a test for AARV. In all other tests, AARV is to be disabled. Four AARV tests are described, at the time constant values Tref = 300s and 5000s, each of those with source voltage steps to (V1+V2)/2 and (V3+V4)/2. Before each voltage step, the steady-state reactive power, Q, should be zero at V=Vref.

The voltage steps result in step changes of Q to 0.5*Q1 and 0.5*Q4, respectively, with some delay from the open-loop response. These new Q values should follow an exponential decay back to zero, governed by Tref = 300s or 5000s. The test criteria specifies that after one time constant, Tref, the value of Q should be less than 10% of Q1 or Q4, respectively. However, the expected value of Q, following an exponential decay from the stepped value, is Q = 0.5*Q1*exp(-Tref/Tref) = 0.1839*Q1 or 0.1839*Q4. **The evaluation criteria at 1*Tref should be 20% instead of 10% of Q1 or Q4**. Note: at 2*Tref, the expected value of Q is 0.0677*Q1 or 0.0677*Q4.

Tref is defined as the time constant for AARV reference voltage. On the other hand, Tresponse is defined as an open-loop response time, which is converted to an exponential time constant in the code below. Furthermore, the simulation is run at a constant time step, *dt*, so both time constants may be implemented as constant decrement factors. If the real controller operates at a constant sample interval, it may also use constant decrement factors to save the time of repeatedly evaluating the exponential functions.

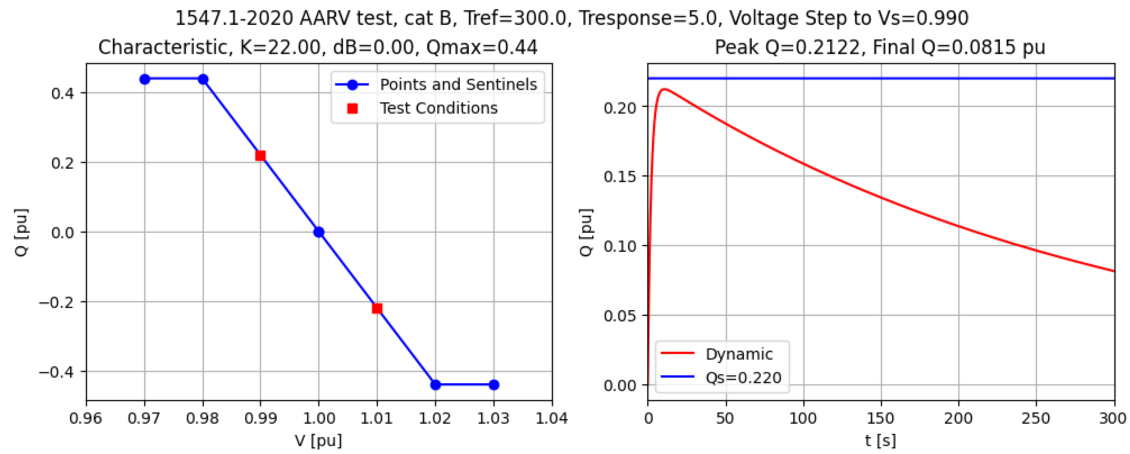
Passing this test verifies implementation of Tref, but not the impact of AARV on a system with grid impedance.

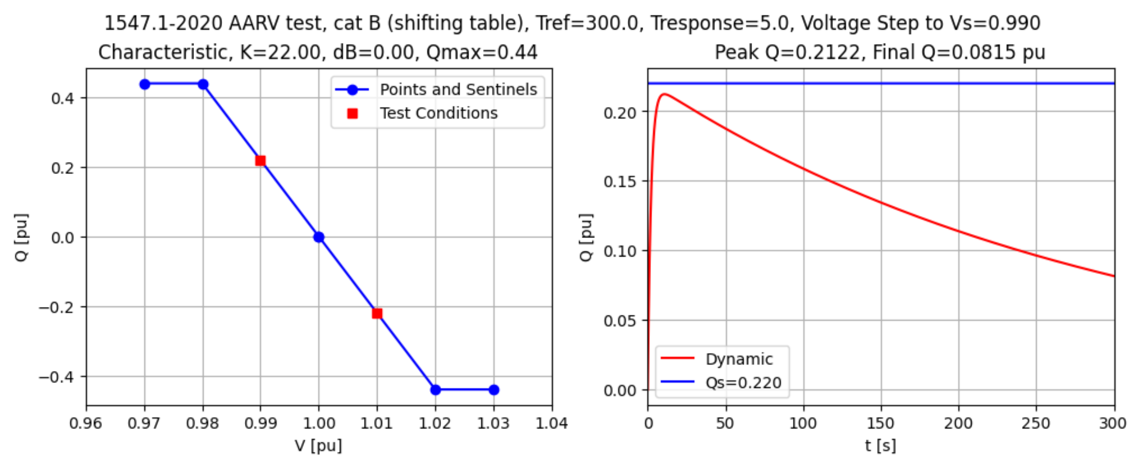
The first example simulates one AARV test, for the step to (V1+V2)/2, and category B with maximum allowed slope and no deadband. The steady-state value of Q at this voltage would be 0.22 pu, but it only reaches 0.2122 pu due to the open-loop response time constant. The value of Q at Tref is 0.0815 pu, which is 18.52% of Q1. (Due to the open-loop response, this is higher than the static expected value of 18.39% at t=Tref). The next three examples show AARV tests for category A (maximum slope), and for category B with minimum and maximum allowed *Tresponse* values.

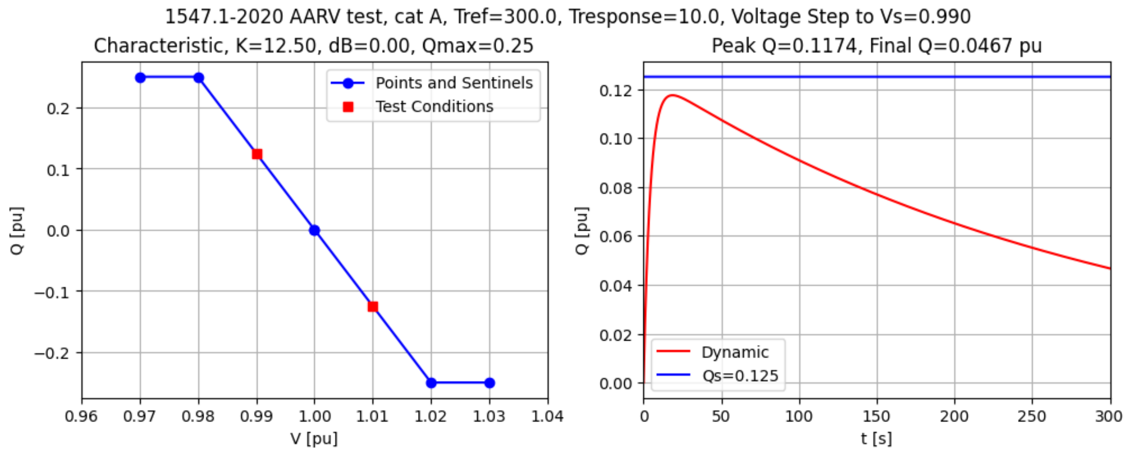
[23]:
```python
def show_aarv_test (tag, Vreg, dB, K, Qmax, Qmin, Tref, Tresponse, dt,␣
 ↪bShiftTable=True):
  TauOL = Tresponse / 2.3026
  IncRef = 1.0 - math.exp(-dt/Tref)
  IncOL = 1.0 - math.exp(-dt/TauOL)
  vtable, qtable = set_characteristic (Vreg, dB, K, Qmax, Qmin)

  vtest = [(vtable[1]+vtable[2])/2.0, (vtable[3]+vtable[4])/2.0]
  qtest = np.interp (vtest, vtable, qtable)

  tmax = 1.0 * Tref
  t = np.linspace (0, tmax, int(tmax / dt) + 1)
  npts = len(t)
  v0 = (vtable[1]+vtable[2])/2.0
  q0 = np.interp(v0, vtable, qtable)
  q = np.zeros(npts)
  vset = 1.0
  vref = 1.0
  qpeak = 0.0
  #print ('        t       Vpu      Vreg      Verr     Qtarg         Q')
  if bShiftTable: # this approach resets the V1..V4 points each time Vref␣
 ↪changes
    for i in range(1, npts):
      verr = v0 - vref
      vref = vref + verr * IncRef
      vtable, qtable = set_characteristic (center=vref, deadband=dB, slope=K,␣
 ↪qmax=Qmax, qmin=Qmin)
      qtarg = np.interp (v0, vtable, qtable)
      q[i] = q[i-1] + (qtarg - q[i-1]) * IncOL
      if abs(q[i]) > qpeak:
        qpeak = abs(q[i])
  else: # this approach modifies the voltage entry point
    for i in range(1, npts):
      verr = v0 - vref
```

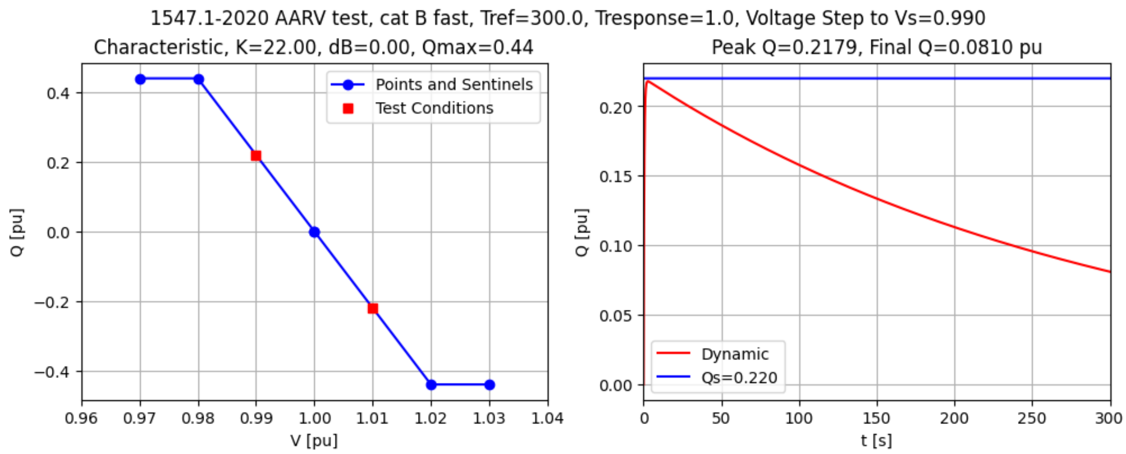
```python
        vref = vref + verr * IncRef
        qtarg = np.interp (v0 + (vset - vref), vtable, qtable)
        q[i] = q[i-1] + (qtarg - q[i-1]) * IncOL
        if abs(q[i]) > qpeak:
          qpeak = abs(q[i])
      #print ('{:8.3f} {:8.5f} {:8.5f} {:8.5f} {:8.5f} {:8.5f}'.format (t[i], v0,␣
 ↪vref, verr, qtarg, q[i]))

#  print ('{:d} test points, IncRef={:.6f}, IncOL={:.6f}, Qpeak={:.3f}, Qend={:.
 ↪3f}'.format (npts, IncRef, IncOL, qpeak, q[-1]))

  fig, ax = plt.subplots(1, 2, sharex = 'col', figsize=(10,4),␣
 ↪constrained_layout=True)
  fig.suptitle ('1547.1-2020 AARV test, {:s}, Tref={:.1f}, Tresponse={:.1f},␣
 ↪Voltage Step to Vs={:.3f}'.format (tag, Tref, Tresponse, v0))

  ax[0].set_title ('Characteristic, K={:.2f}, dB={:.2f}, Qmax={:.2f}'.format␣
 ↪(K, dB, Qmax))
  # remove any shift of vtable, qtable that might have been done while␣
 ↪simulating the test
  vtable, qtable = set_characteristic (Vreg, dB, K, Qmax, Qmin)
  ax[0].plot (vtable, qtable, marker='o', color='blue', label='Points and␣
 ↪Sentinels')
  ax[0].plot (vtest, qtest, color='red', marker='s', linestyle='None',␣
 ↪label='Test Conditions')
  ax[0].set_xlabel ('V [pu]')
  ax[0].set_xlim (vtable[0]-0.01, vtable[-1]+0.01)
  ax[0].legend()

  ax[1].set_title ('Peak Q={:.4f}, Final Q={:.4f} pu'.format (qpeak, q[-1]))
  ax[1].plot (t, q, color='red', label='Dynamic')
  ax[1].plot (t, q0*np.ones(npts), color='blue', label='Qs={:.3f}'.format(q0))
  ax[1].set_xlabel ('t [s]')
  ax[1].set_xlim (t[0], t[-1])

  for i in range(2):
    ax[1].legend()
    ax[i].grid()
    ax[i].set_ylabel ('Q [pu]')

  plt.show()

show_aarv_test ('cat B', Vreg=1, dB=0, K=22, Qmax=0.44, Qmin=-0.44, Tref=300,␣
 ↪Tresponse=5, dt=0.1)  # for most aggressive category B
```

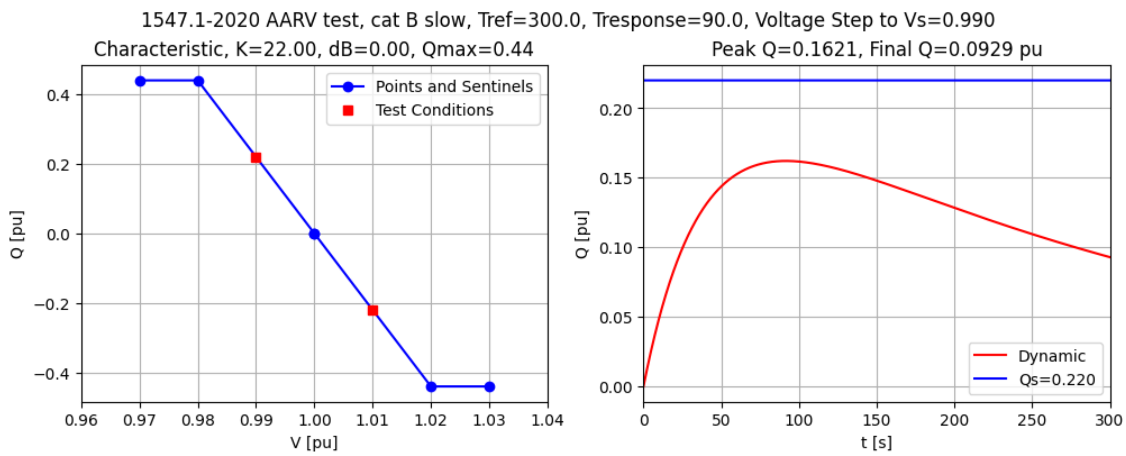
```
show_aarv_test ('cat B (shifting table)', Vreg=1, dB=0, K=22, Qmax=0.44,␣
↪Qmin=-0.44, Tref=300, Tresponse=5, dt=0.1, bShiftTable=True)  # for most␣
↪aggressive category B
show_aarv_test ('cat A', Vreg=1, dB=0, K=12.5, Qmax=0.25, Qmin=-0.25, Tref=300,␣
↪Tresponse=10, dt=0.1) # for most aggressive category A
show_aarv_test ('cat B fast', Vreg=1, dB=0, K=22, Qmax=0.44, Qmin=-0.44,␣
↪Tref=300, Tresponse=1, dt=0.1)  # category B with lowest Tresponse
show_aarv_test ('cat B slow', Vreg=1, dB=0, K=22, Qmax=0.44, Qmin=-0.44,␣
↪Tref=300, Tresponse=90, dt=0.1) # category B with highest Tresponse
```






1547.1-2020 AARV test, cat A, Tref=300.0, Tresponse=10.0, Voltage Step to Vs=0.990
Characteristic, K=12.50, dB=0.00, Qmax=0.25
Points and Sentinels
Test Conditions
Q [pu]
V [pu]
Peak Q=0.1174, Final Q=0.0467 pu
Dynamic
Qs=0.125
Q [pu]
t [s]

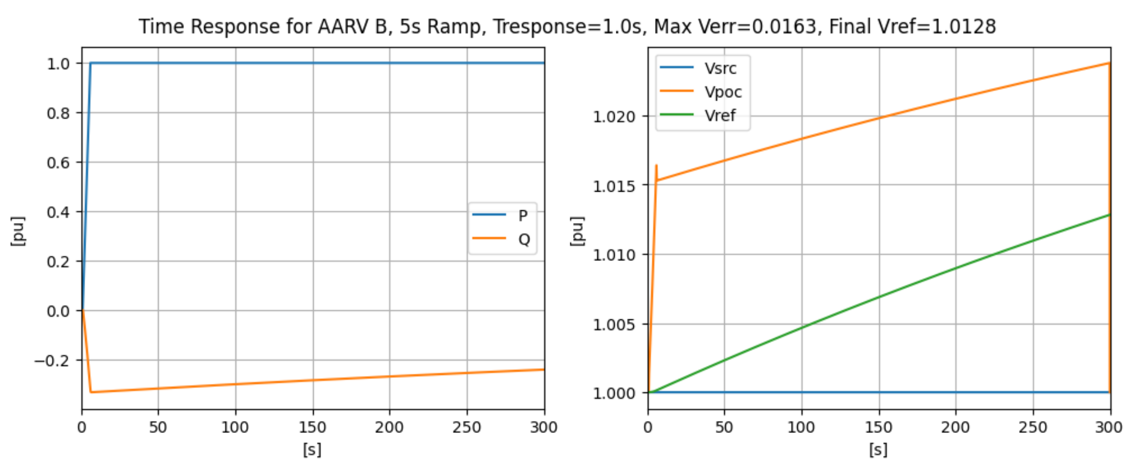

1547.1-2020 AARV test, cat B fast, Tref=300.0, Tresponse=1.0, Voltage Step to Vs=0.990
Characteristic, K=22.00, dB=0.00, Qmax=0.44
Points and Sentinels
Test Conditions
Q [pu]
V [pu]
Peak Q=0.2179, Final Q=0.0810 pu
Dynamic
Qs=0.220
Q [pu]
t [s]

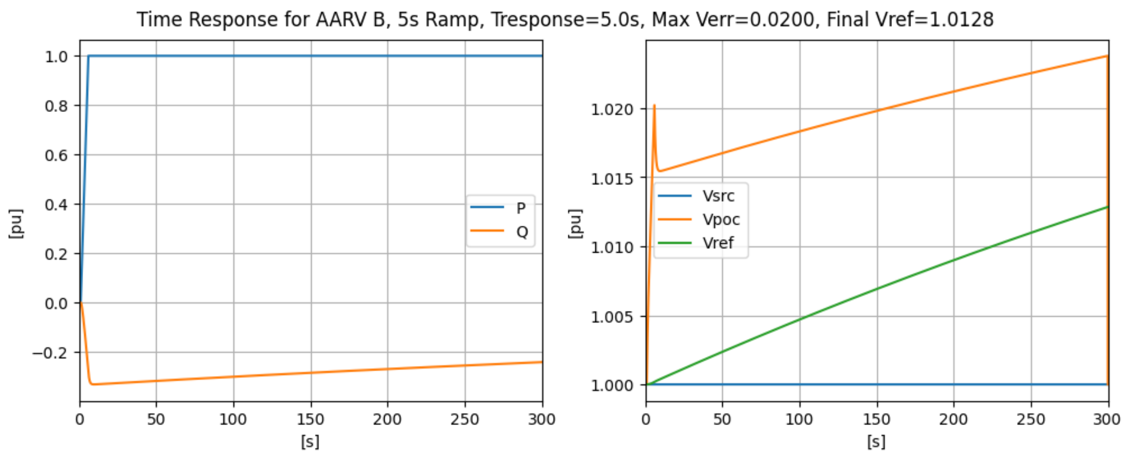

1547.1-2020 AARV test, cat B slow, Tref=300.0, Tresponse=90.0, Voltage Step to Vs=0.990
Characteristic, K=22.00, dB=0.00, Qmax=0.44
Points and Sentinels
Test Conditions
Q [pu]
V [pu]
Peak Q=0.1621, Final Q=0.0929 pu
Dynamic
Qs=0.220
Q [pu]
t [s]

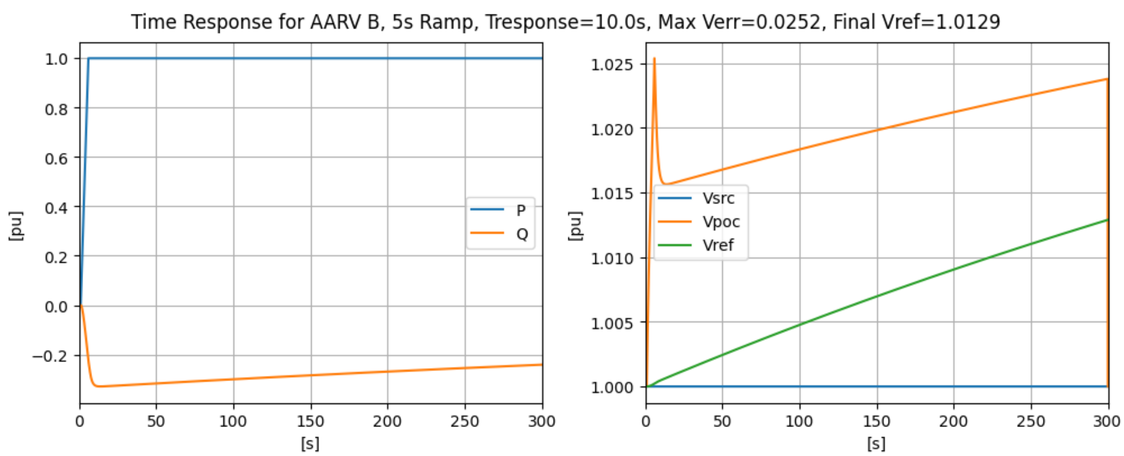
0.4 AARV Response to Storage Power-On Ramping

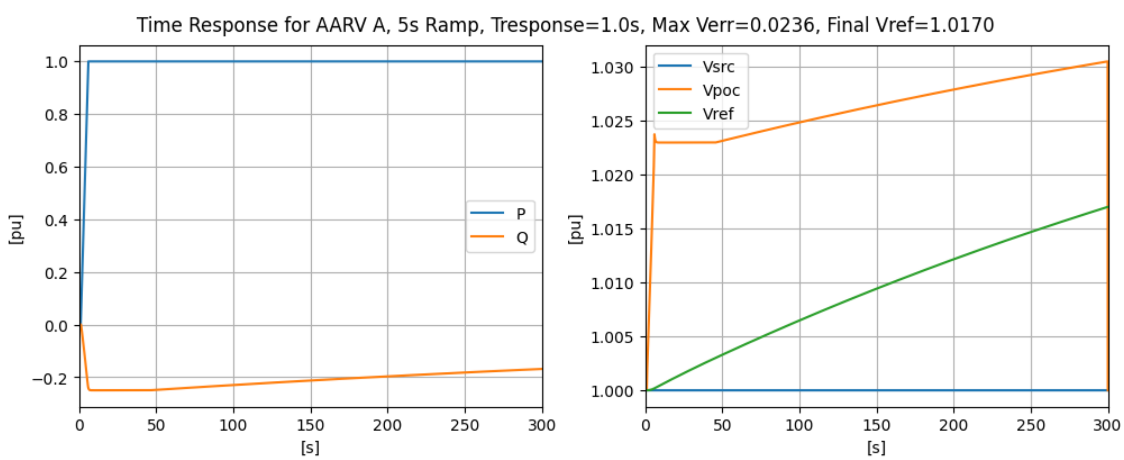
[24]:
```python
def show_bess_test (tag, Vreg, dB, K, Qmax, Qmin, Tref, Tresponse, dt=0.1,␣
 ↪numTrefs=1, RampTime=5, bShiftTable=False):
  TauOL = Tresponse / 2.3026
  if Tref > 0.0:
    IncRef = 1.0 - math.exp(-dt/Tref)
    tmax = numTrefs * Tref
  else:
    IncRef = 0.0
    tmax = 300.0
  IncOL = 1.0 - math.exp(-dt/TauOL)
  vtable, qtable = set_characteristic (Vreg, dB, K, Qmax, Qmin)

  t = np.linspace (0, tmax, int(tmax / dt) + 1)
  npts = len(t)

  vsrc = np.ones(npts)
  vpoc = np.ones(npts)
  pstart = 1.0
  pend = pstart + RampTime
  p = np.interp(t, [0.0, pstart, pend, t[-1]], [0.0, 0.0, DP, DP])
  q = np.zeros(npts)
  vref = np.ones(npts)
  verr = np.zeros(npts)
  # this approach resets the V1..V4 points each time Vref changes
  for i in range(npts-1):
    vpoc[i], d = get_voltage (RPU, XPU, p[i], q[i], vsrc[i])
    verr[i] = vpoc[i] - vref[i]
    vref[i+1] = vref[i] + verr[i] * IncRef
    if bShiftTable: # this approach resets the V1..V4 points each time Vref␣
 ↪changes
      vtable, qtable = set_characteristic (vref[i], deadband=dB, slope=K,␣
 ↪qmax=Qmax, qmin=Qmin)
      qtarg = np.interp (vpoc[i], vtable, qtable)
    else: # this approach modifies the voltage entry point
      qtarg = np.interp (Vreg + verr[i], vtable, qtable)
    q[i+1] = q[i] + (qtarg - q[i]) * IncOL

  fig, ax = plt.subplots(1, 2, sharex = 'col', figsize=(10,4),␣
 ↪constrained_layout=True)
  fig.suptitle ('Time Response for {:s}, Max Verr={:.4f}, Final Vref={:.4f}'.
 ↪format (tag, np.max(verr), vref[-1]))
  ax[0].plot (t, p, label='P')
  ax[0].plot (t, q, label='Q')
  ax[1].plot (t, vsrc, label='Vsrc')
  ax[1].plot (t, vpoc, label='Vpoc')
```

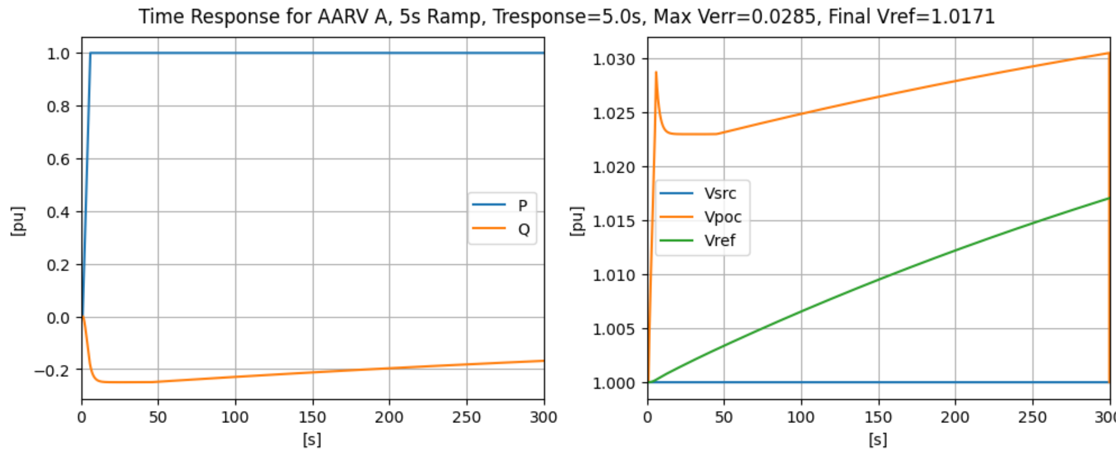
```
  ax[1].plot (t, vref, label='Vref')
  #ax[1].plot (t, verr, label='Verr')
  for i in range(2):
    ax[i].set_ylabel ('[pu]')
    ax[i].set_xlabel ('[s]')
    ax[i].set_xlim (t[0], t[-1])
    ax[i].legend()
    ax[i].grid()
  plt.show()

for tr in [1.0, 5.0, 10.0]:
  tag = 'AARV B, 5s Ramp, Tresponse={:.1f}s'.format (tr)
  show_bess_test (tag, Vreg=1, dB=0, K=22, Qmax=0.44, Qmin=-0.44, Tref=300,␣
 ↪Tresponse=tr, dt=0.1, numTrefs=1.0, RampTime=5.0)

for tr in [1.0, 5.0, 10.0]:
  tag = 'AARV A, 5s Ramp, Tresponse={:.1f}s'.format (tr)
  show_bess_test (tag, Vreg=1, dB=0, K=12.5, Qmax=0.25, Qmin=-0.25, Tref=300,␣
 ↪Tresponse=tr, dt=0.1, numTrefs=1.0, RampTime=5.0)

#show_bess_test ('5s Ramp, No Volt-Var', Vreg=1, dB=0, K=22, Qmax=0, Qmin=0,␣
 ↪Tref=300, Tresponse=5, dt=0.1, numTrefs=1.0, RampTime=5.0, bShiftTable=True)
#show_bess_test ('5s Ramp, No Volt-Var', Vreg=1, dB=0, K=22, Qmax=0, Qmin=0,␣
 ↪Tref=300, Tresponse=5, dt=0.1, numTrefs=1.0, RampTime=5.0, bShiftTable=False)
#show_bess_test ('Q decay', Vreg=1, dB=0, K=22, Qmax=0.44, Qmin=-0.44,␣
 ↪Tref=300, Tresponse=5, dt=0.1, numTrefs=10.0, RampTime=5.0, bShiftTable=True)
#show_bess_test ('Q decay', Vreg=1, dB=0, K=22, Qmax=0.44, Qmin=-0.44,␣
 ↪Tref=300, Tresponse=5, dt=0.1, numTrefs=10.0, RampTime=5.0,␣
 ↪bShiftTable=False)
```

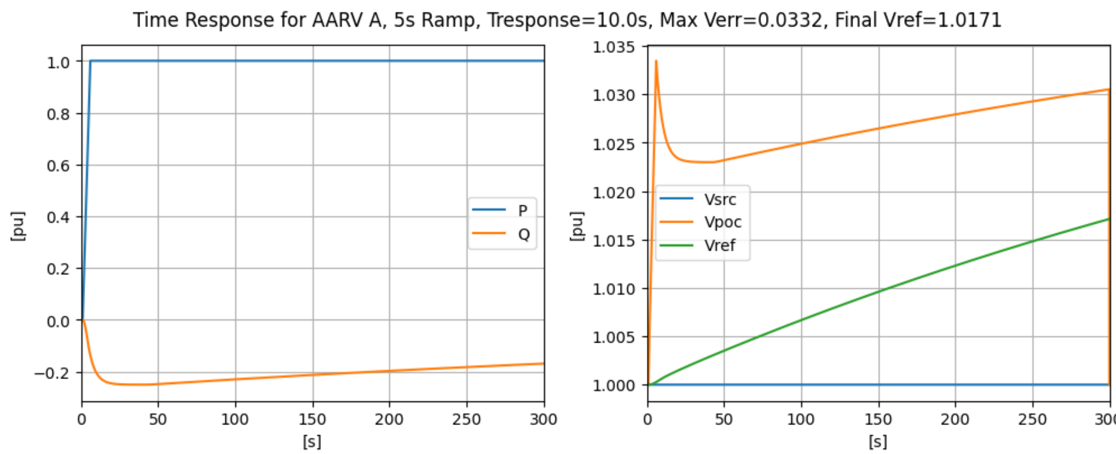



Time Response for AARV B, 5s Ramp, Tresponse=5.0s, Max Verr=0.0200, Final Vref=1.0128
P
Q
Vsrc
Vpoc
Vref
[pu]
[s]


Time Response for AARV B, 5s Ramp, Tresponse=10.0s, Max Verr=0.0252, Final Vref=1.0129
P
Q
Vsrc
Vpoc
Vref
[pu]
[s]


Time Response for AARV A, 5s Ramp, Tresponse=1.0s, Max Verr=0.0236, Final Vref=1.0170
P
Q
Vsrc
Vpoc
Vref
[pu]
[s]



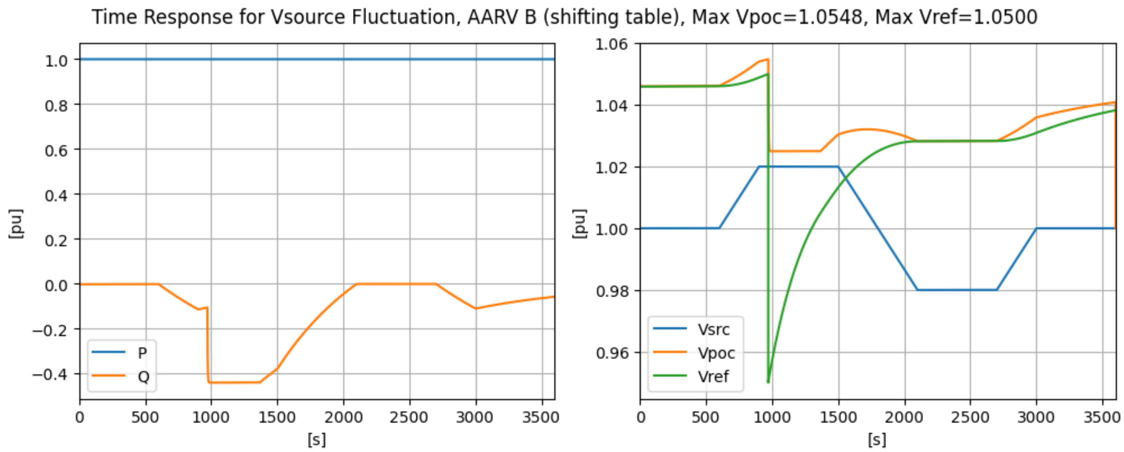


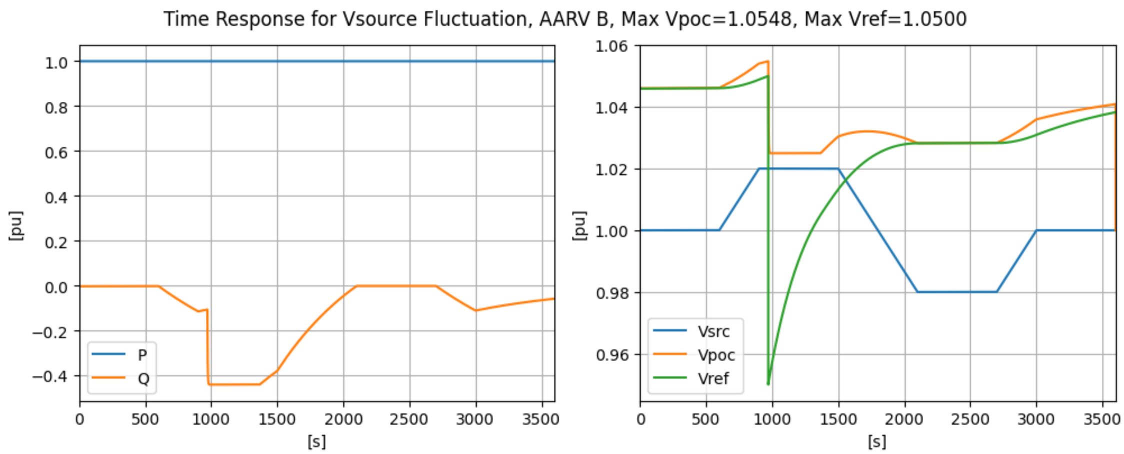
0.5 AARV Response to Grid Voltage Fluctuations

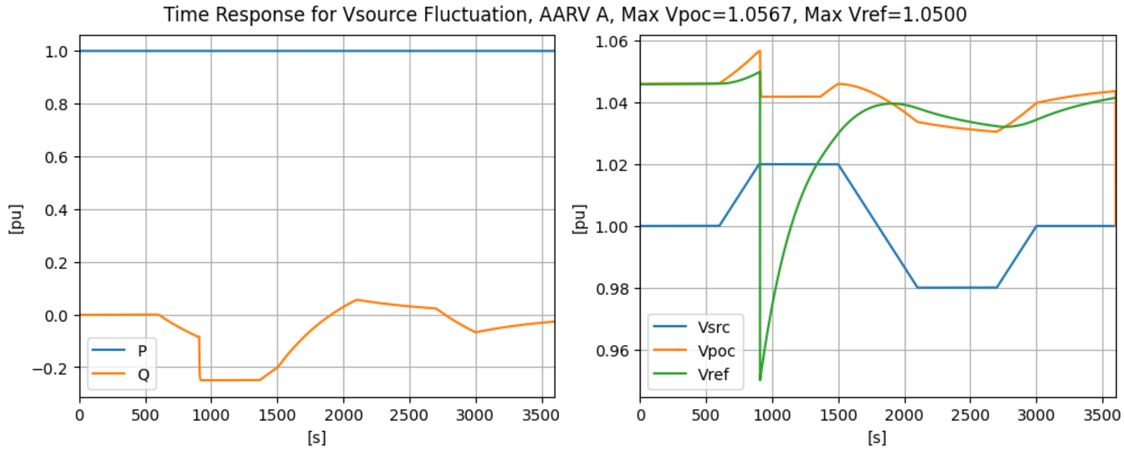
[25]:
```python
def show_vsrc_test (tag, Vreg, dB, K, Qmax, Qmin, Tref, Tresponse, dt=0.1,
 ↪Vref=1.0, bShiftTable=False):
  TauOL = Tresponse / 2.3026
  if Tref > 0.0:
    IncRef = 1.0 - math.exp(-dt/Tref)
  else:
    IncRef = 0.0
    Tref = 300.0
  IncOL = 1.0 - math.exp(-dt/TauOL)
  vtable, qtable = set_characteristic (Vreg, dB, K, Qmax, Qmin)

  tmax = 12.0 * Tref
  t = np.linspace (0, tmax, int(tmax / dt) + 1)
  npts = len(t)
```

```
 vsrc = np.interp (t, [0.0, 2*Tref, 3*Tref, 5*Tref, 7*Tref, 9*Tref, 10*Tref,␣
↪t[-1]],
                          [1.0, 1.0,    1.02,   1.02,   0.98,   0.98,   1.0,     1.
↪0])
 vpoc = np.ones(npts)
 p = DP * np.ones(npts)
 q = np.zeros(npts)
 vref = Vref * np.ones(npts)
 verr = np.zeros(npts)
 for i in range(npts-1):
   vpoc[i], d = get_voltage (RPU, XPU, p[i], q[i], vsrc[i])
   verr[i] = vpoc[i] - vref[i]
   vref[i+1] = vref[i] + verr[i] * IncRef
   if vref[i+1] > 1.05:
     vref[i+1] = 0.95
   if bShiftTable: # this approach resets the V1..V4 points each time Vref␣
↪changes
     vtable, qtable = set_characteristic (vref[i], deadband=dB, slope=K,␣
↪qmax=Qmax, qmin=Qmin)
     qtarg = np.interp (vpoc[i], vtable, qtable)
   else: # this approach modifies the voltage entry point
     qtarg = np.interp (Vreg + verr[i], vtable, qtable)
   q[i+1] = q[i] + (qtarg - q[i]) * IncOL

 fig, ax = plt.subplots(1, 2, sharex = 'col', figsize=(10,4),␣
↪constrained_layout=True)
 fig.suptitle ('Time Response for {:s}, Max Vpoc={:.4f}, Max Vref={:.4f}'.
↪format (tag, np.max(vpoc), np.max(vref)))
 ax[0].plot (t, p, label='P')
 ax[0].plot (t, q, label='Q')
 ax[1].plot (t, vsrc, label='Vsrc')
 ax[1].plot (t, vpoc, label='Vpoc')
 ax[1].plot (t, vref, label='Vref')
 #ax[1].plot (t, verr, label='Verr')
 for i in range(2):
   ax[i].set_ylabel ('[pu]')
   ax[i].set_xlabel ('[s]')
   ax[i].set_xlim (t[0], t[-1])
   ax[i].legend()
   ax[i].grid()
 plt.show()

show_vsrc_test ('Vsource Fluctuation, AARV B (shifting table)', Vreg=1, dB=0,␣
↪K=22, Qmax=0.44, Qmin=-0.44, Tref=300, Tresponse=5, dt=0.1, Vref=1.0459,␣
↪bShiftTable=True)
```

```
show_vsrc_test ('Vsource Fluctuation, AARV B', Vreg=1, dB=0, K=22, Qmax=0.44,␣
    ↪Qmin=-0.44, Tref=300, Tresponse=5, dt=0.1, Vref=1.0459, bShiftTable=False)
show_vsrc_test ('Vsource Fluctuation, AARV A', Vreg=1, dB=0, K=12.5, Qmax=0.25,␣
    ↪Qmin=-0.25, Tref=300, Tresponse=5, dt=0.1, Vref=1.0459)
```

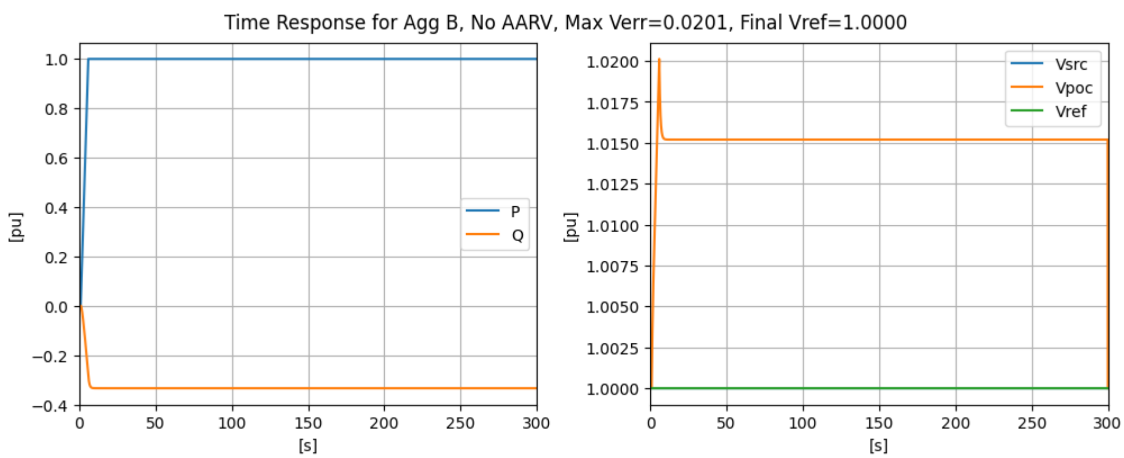


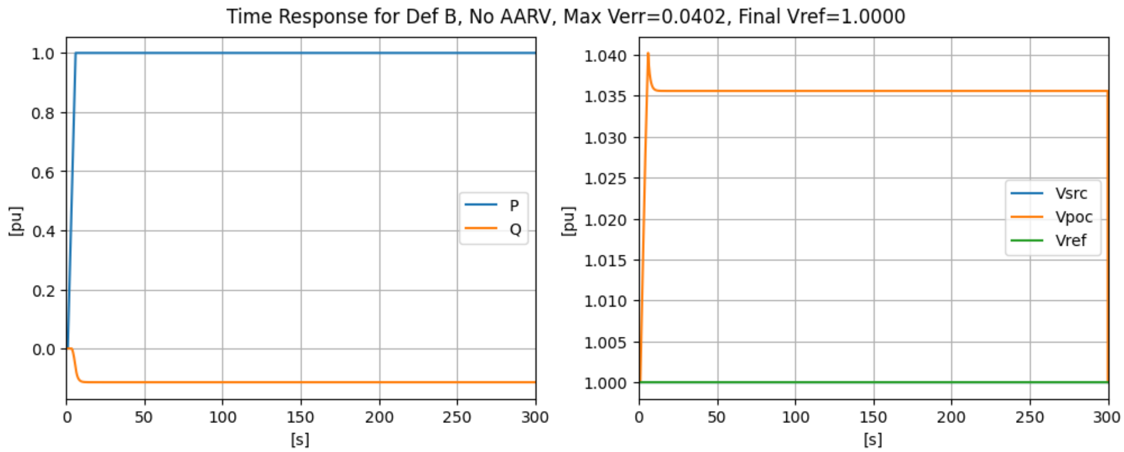


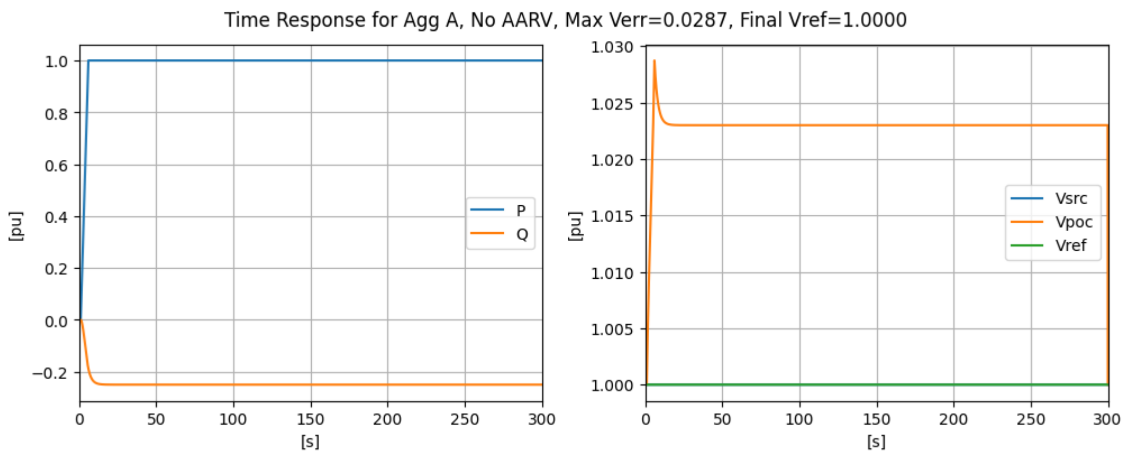


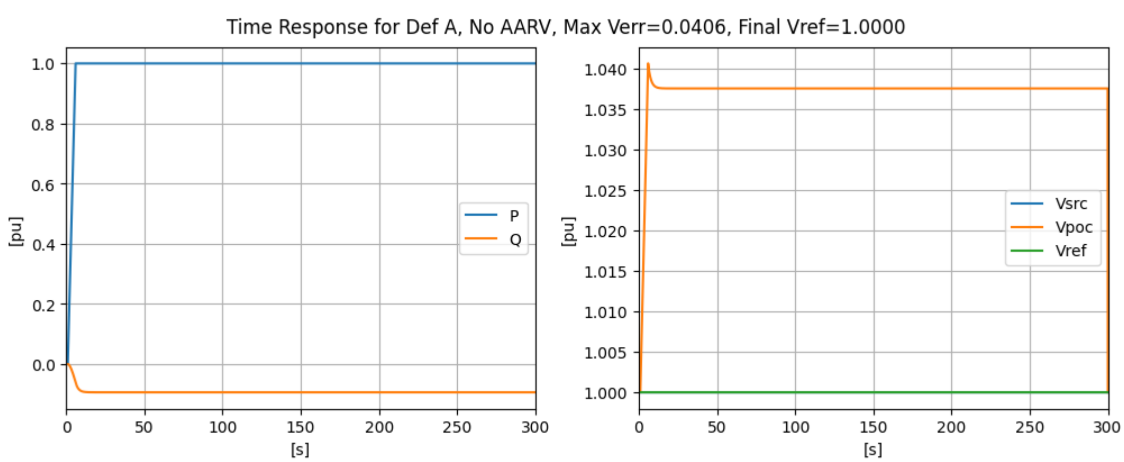
0.6 Volt-Var Response to Storage Power-On Ramping

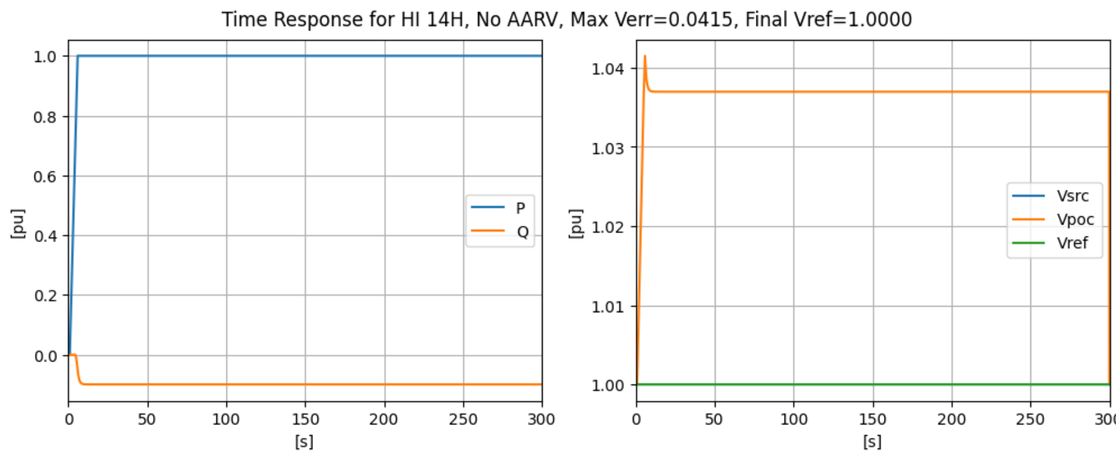
[26]:
```
show_bess_test ('Agg B, No AARV', Vreg=1, dB=0, K=22, Qmax=0.44, Qmin=-0.44,␣
    ↪Tref=0, Tresponse=5)
show_bess_test ('Def B, No AARV', Vreg=1, dB=0.04, K=22.0/3.0, Qmax=0.44,␣
    ↪Qmin=-0.44, Tref=0, Tresponse=5)
show_bess_test ('Agg A, No AARV', Vreg=1, dB=0, K=12.5, Qmax=0.25, Qmin=-0.25,␣
    ↪Tref=0, Tresponse=5)
show_bess_test ('Def A, No AARV', Vreg=1, dB=0, K=2.50, Qmax=0.25, Qmin=-0.25,␣
    ↪Tref=0, Tresponse=5)
show_bess_test ('HI 14H, No AARV', Vreg=1, dB=0.06, K=43.0/3.0, Qmax=0.44,␣
    ↪Qmin=-0.44, Tref=0, Tresponse=5)
```




Time Response for Def B, No AARV, Max Verr=0.0402, Final Vref=1.0000
[pu]
P
Q
[s]
Vsrc
Vpoc
Vref

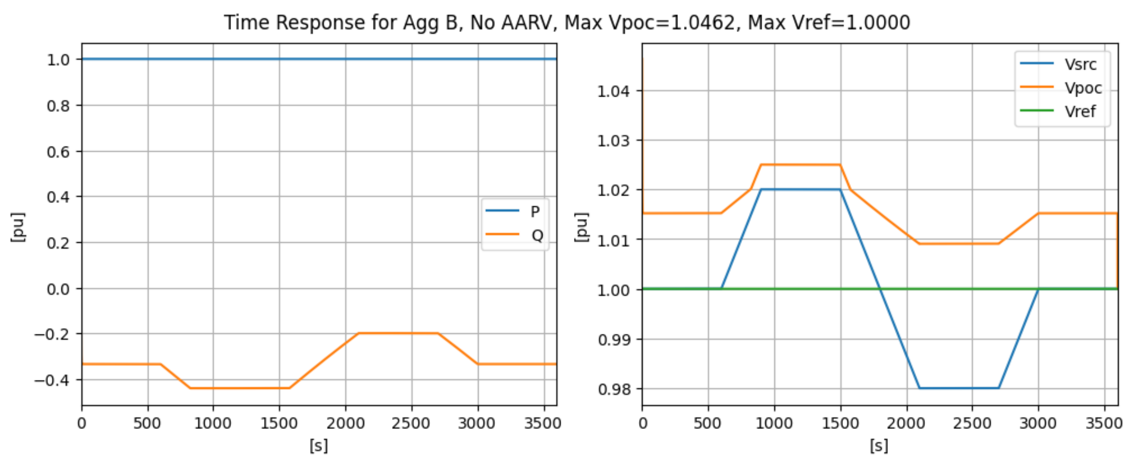

Time Response for Agg A, No AARV, Max Verr=0.0287, Final Vref=1.0000
[pu]
P
Q
[s]
Vsrc
Vpoc
Vref

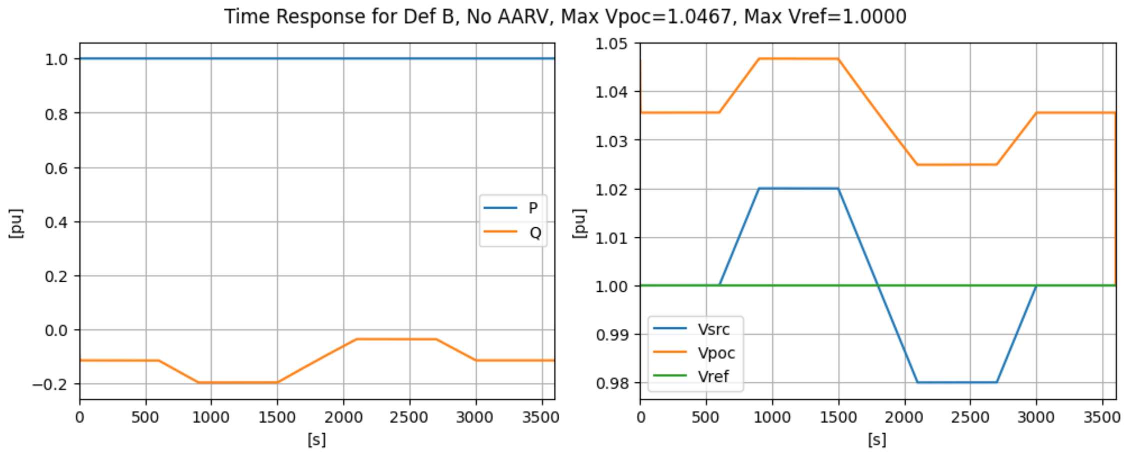

Time Response for Def A, No AARV, Max Verr=0.0406, Final Vref=1.0000
[pu]
P
Q
[s]
Vsrc
Vpoc
Vref

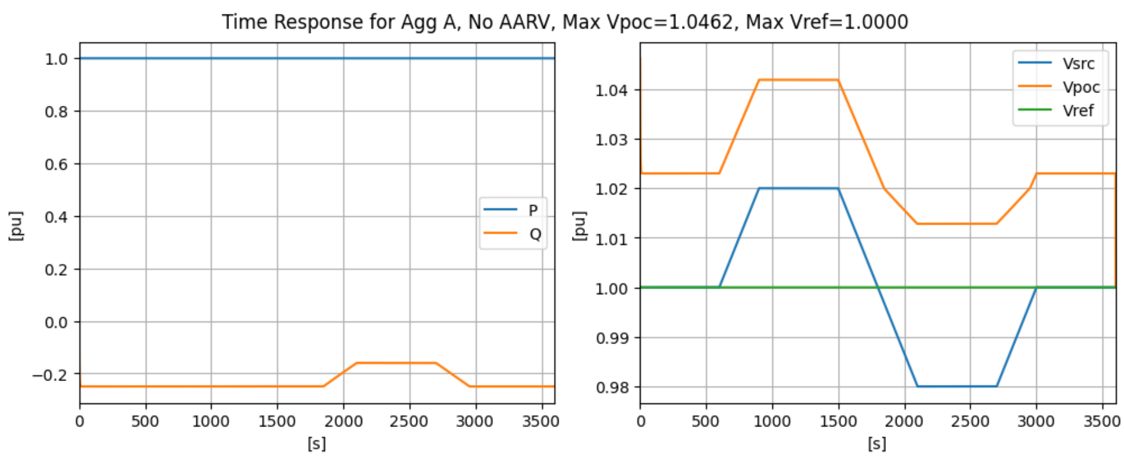


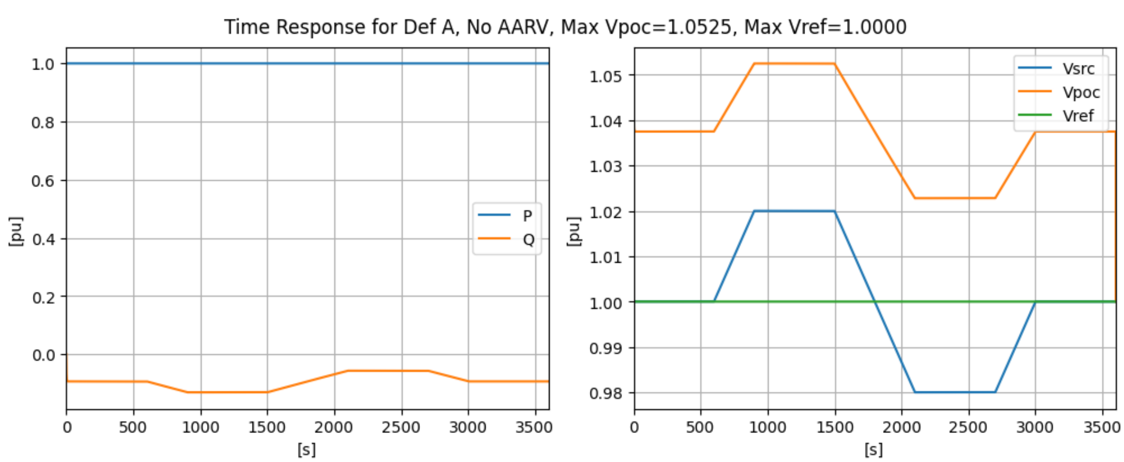
0.7 Volt-Var Response to Grid Voltage Fluctuations

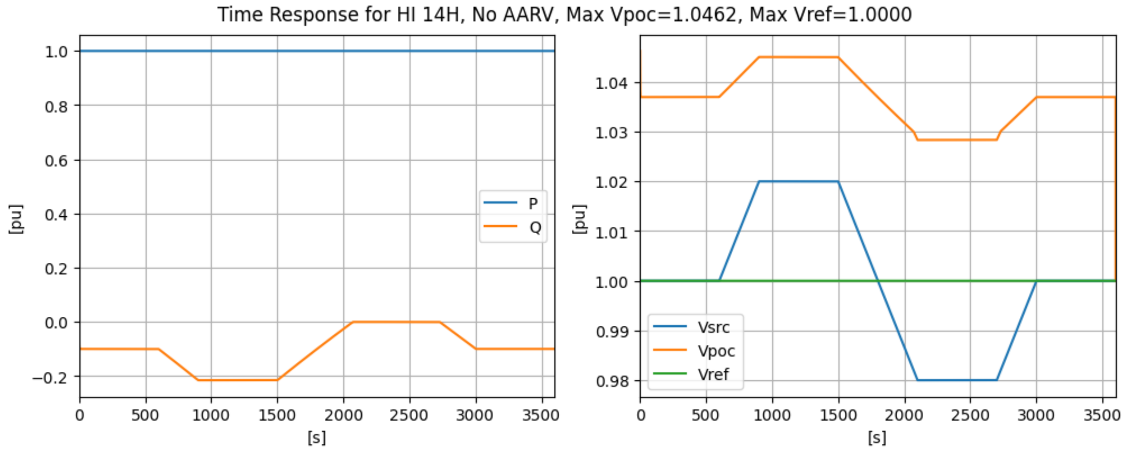
[27]:
```
show_vsrc_test ('Agg B, No AARV', Vreg=1, dB=0, K=22, Qmax=0.44, Qmin=-0.44,␣
    ↪Tref=0, Tresponse=5)
show_vsrc_test ('Def B, No AARV', Vreg=1, dB=0.04, K=22.0/3.0, Qmax=0.44,␣
    ↪Qmin=-0.44, Tref=0, Tresponse=5)
show_vsrc_test ('Agg A, No AARV', Vreg=1, dB=0, K=12.5, Qmax=0.25, Qmin=-0.25,␣
    ↪Tref=0, Tresponse=5)
show_vsrc_test ('Def A, No AARV', Vreg=1, dB=0, K=2.50, Qmax=0.25, Qmin=-0.25,␣
    ↪Tref=0, Tresponse=5)
show_vsrc_test ('HI 14H, No AARV', Vreg=1, dB=0.06, K=43.0/3.0, Qmax=0.44,␣
    ↪Qmin=-0.44, Tref=0, Tresponse=5)
```




Time Response for Def B, No AARV, Max Vpoc=1.0467, Max Vref=1.0000
[pu]
[s]
P
Q
Vsrc
Vpoc
Vref


Time Response for Agg A, No AARV, Max Vpoc=1.0462, Max Vref=1.0000
[pu]
[s]
P
Q
Vsrc
Vpoc
Vref


Time Response for Def A, No AARV, Max Vpoc=1.0525, Max Vref=1.0000
[pu]
[s]
P
Q
Vsrc
Vpoc
Vref


Time Response for HI 14H, No AARV, Max Vpoc=1.0462, Max Vref=1.0000
[pu]
[s]
P
Q
Vsrc
Vpoc
Vref

```
[ ]:
```