

Load Characteristic LD1PAC

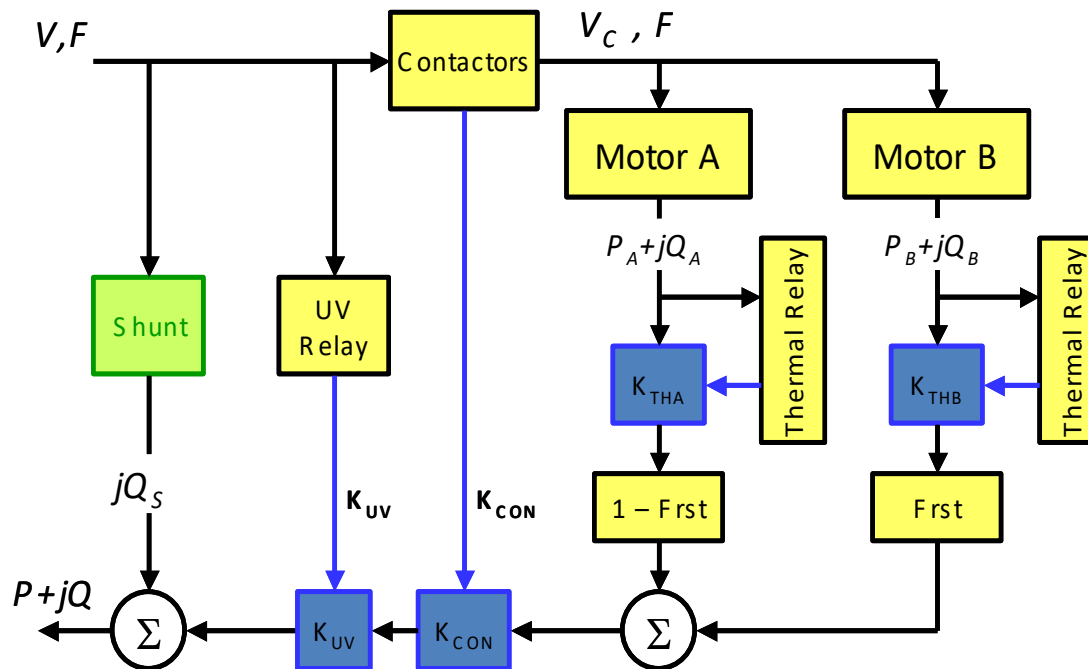
Load Characteristic Model LD1PAC

Pul	Fraction of constant power load
TV	Voltage input time in sec.
Tf	Frequency input time constant in sec.
CompPF	Compressor Power Factor
Vstall	Compressor Stalling Voltage in p.u.
Rstall	Compressor Stall resistance in p.u.
Xstall	Compressor Stall impedance in p.u.
Tstall	Compressor Stall delay time in sec.
LFadj	Vstall adjustment proportional to loading factor
KP1 to KP2	Real power coefficient for running states, p.u.P/p.u.V
NP1 to NP2	Real power exponent for running states
KQ1 to KQ2	Reactive power coefficient for running states, p.u.Q/p.u.
NQ1 to NQ2	Reactive power exponent for running states
Vbrk	Compressor motor breakdown voltage in p.u
Frst	Restarting motor fraction
Vrst	Restart motor voltage in p.u.
Trst	Restarting time delay in sec.
CmpKpf	Real power frequency sensitivity, p.u.P/p.u.
CmpKqf	Reactive power frequency sensitivity, p.u.Q/p.u.

Vc1off	Voltage 1 contactor disconnect load in p.u.
Vc2off	Voltage 2 contactor disconnect load in p.u.
Vc1on	Voltage 1 contactor re-connect load in p.u.
Vc2on	Voltage 2 contactor re-connect load in p.u.
Tth	Compressor heating time constant in sec.
Th1t	Compressor motors begin tripping
Th2t	Compressor motors finished tripping
fuvr	Fraction of compressor motors with undervoltage relays
uvtr1 to uvtr2	Undervoltage pickup level in p.u.
ttr1 to ttr2	Undervoltage definite time in sec.
MBase	MVABase. Values of zero and negative have special interpretation described below

Model supported by PSLF

LD1PAC represents a composite of many individual single-phase A/C compressors and their protective devices as shown below.



The compressor motor model is divided into two parts:

Motor A – Those compressors that can't restart soon after stalling

Motor B – Those compressors that can restart soon after stalling

If $Mbase > 0$ then this value of $MVABase$ is used and the $CompLF = 1.0$

If $Mbase = 0$ then $MVABase = Pinit * Pul$ and $CompLF = 1.0$

If $Mbase < 0$ then $CompLF = abs(Mbase)$ and $MVABase = Pinit * Pul / CompLF$

The values of $Vstall$ and $Vbrk$ are adjusted according to the value of $LFAdj$.

$$Vstall = Vstall[1 + LFAdj(CompLF - 1)]$$

$$Vbrk = Vbrk[1 + LFAdj(CompLF - 1)]$$

Algebraic P and Q Model

If $V < V_{stall}$ for more than T_{stall} seconds, then the motor will enter the stall state and remain on the P and Q curves defined by

$$P = G_{stall} * V^2$$

$$Q = B_{stall} * V^2$$

For the “B” motor, if $V > V_{rst}$ for more than T_{rst} second, then the motor will restart and automatically flip back to the curves as follows. [Special treatment of $T_{sall} < 0$ was added in October 16, 2019 patch of Simulator 20 and 21. See note at end.]

While the stall timer (T_{stall}) has not been exceeded however, the motors are represented by algebraic equations, as follows:

If $V > V_{brk}$

$$P = [P_o + K_{p1}(V - V_{brk})^{N_{p1}}][1 + CmpKpf * \Delta f]$$

$$Q = [Q_o + K_{q1}(V - V_{brk})^{N_{q1}}][1 + CmpKqf * \Delta f]$$

If $V < V_{brk}$ and $V > V_{stallbrk}$

$$P = [P_o + K_{p2}(V_{brk} - V)^{N_{p2}}][1 + CmpKpf * \Delta f]$$

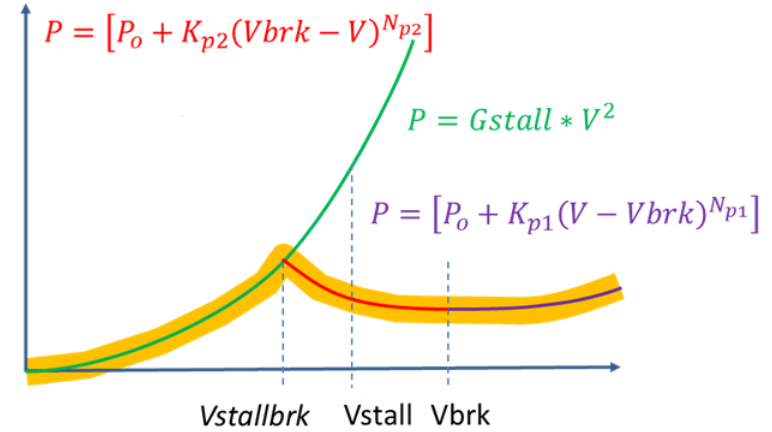
$$Q = [Q_o + K_{q2}(V_{brk} - V)^{N_{q2}}][1 + CmpKqf * \Delta f]$$

If $V < V_{stallbrk}$:

$$P = G_{stall} * V^2$$

$$Q = B_{stall} * V^2$$

The P and Q values are in per unit on the MVABase calculated on the previous page.



This is generally depicted in the figures to the right where $V_{stallbrk}$, P_o , and Q_o are calculated during initialization.

From the initial terminal voltage, power and reactive power (V_{init} , P_{init} , Q_{init}), the following steps are taken to initialize this model

1. $P_o = P_{init} - K_{p1}(V_{init} - V_{brk})^{N_{p1}}$
2. $V_{stallbrk}$ is automatically calculated by determining the intersection of the power stall curve and the power curve defined by K_{p2} and N_{p2} . (PowerWorld determines this to a tolerance of 0.0001 per unit voltage).

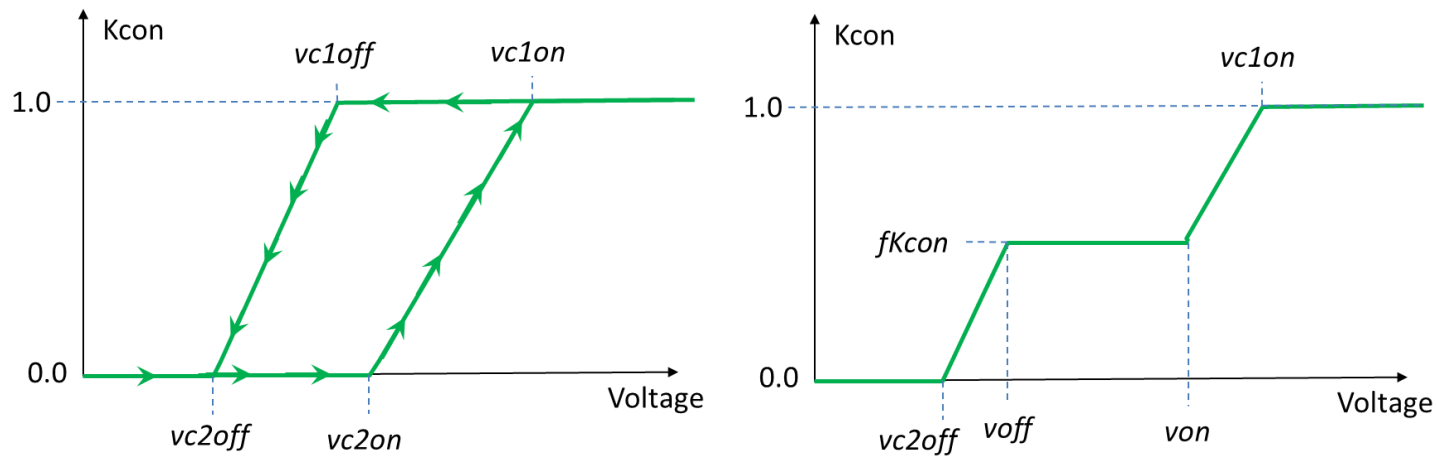
Note: Prior to the October 7, 2016 Simulator, if this intersection was calculated as higher than V_{stall} , then we instead set $V_{stallbrk} = V_{stall}$. This was discontinued on October 7, 2016 and later patches based on user feedback. Setting V_{stall} to small values was not expected when this model was created, but users were setting V_{stall} to very small voltages as a way of instructing the model to never go into the stall curve, resulting in simulation results operating very high on the red curve above which were not valid.

3. $Q_o = P_{init} \left(\frac{\sqrt{1 - CompPF^2}}{CompPF} \right) - K_{p1}(1.0 - V_{brk})^{N_{p1}}$

When initializing reactive power that is not accounted for by this initialization will be treated using the static load model assigned to the load.

Contactor Model

The contactor model represents a model of how air conditioners trip off as the voltage falls and then turn back on as the voltage increases. They start tripping off at a voltage of $vc1off$ and then fall off linearly until all are completely off at $vc2off$. They then start turning back on at $vc2on$ and increase linearly until all are back on at $vc1on$. This is shown on the left curve below. To be more explicit regarding the way this is handled in the code though, at the beginning of each time step a value of $Kcon$ is calculated and values for $voff$ and von at which the respective curves are intercepted. Once $Kcon$ drops below 1.0 then as the voltage moves back up and down it will move at a constant $Kcon$ level until it hits the other line as shown in the figure on the right.



Under Voltage Relay

If the voltage drops below $uvtr1$ for $ttr1$ seconds, or below $uvtr2$ for $ttr2$ seconds, then a fraction of the load ($fuvtr$) will be tripped. Once this fraction is tripped it will not be reconnected.

Thermal overload protection

A motor will calculate the value I^2R_{stall} (values calculated in per unit on the MVA base of that portion of the load) and run this signal through a thermal time constant of T_{th} . A separate temperature is maintained for the “A” and “B” portions of the load. If the temperature is below $Th1t$, then it remains in service load, but starting at $Th1t$ a fraction of the load will be tripped and the fraction tripped will linearly increase up until at a temperature of $Th2t$ at which all of that portion of the load is tripped.

Stalling Transition using Inverse-Time Characteristic

[Special situation of entering *Tstall* as a negative value was add in October 16, 2019 patch of Simulator 20 and 21]

Normally if $V < V_{stall}$ for more than T_{stall} seconds, then the motor will enter the stall state. This represents a *definite* time transition to the stall state. If however the T_{stall} value is entered as a negative number, then a special hard-coded *inverse* time characteristic is used instead. This was discussed with the NERC LMTF meetings in October 2019 and the lookup table for T_{stall} based on terminal voltage is shown to the right. This inverse time characteristic is updated at the end of time step in the numerical integration. The follow pseudo-code represents what is done.

```
// Variables that will be stored with this model
// StallVoltTimeSet = Flag to indicate if we have set the LastStallVoltTime yet
// LastStallVoltTime = Time of previous time step
// LastTCycles = store (1/local_TCycles) from previous time step
// StallState = algebraic "state" that is integrating up to 1.0 to indicate stalled
// Variables we get from the simulation
// PresentTime = Present time
// PresentV = the present terminal voltage of the LD1PAC model
// Local Variable
// local_TCycles = local variable that stores the lookup table result for the Present V

if PresentV >= 0.565 then begin
  if StallVoltTimeSet then begin
    // write message indicating the stall time has cleared if had previously been integrating
    end
    StallVoltTimeSet = False
    StallState = 0 // reset
  end
else begin
  if PresentV < 0.450 then local_TCycles = 2
  else if PresentV < 0.490 then local_TCycles = 2 + (3-2)*(PresentV - 0.450)/(0.490-0.450)
  else if PresentV < 0.550 then local_TCycles = 3 + (12-3)*(PresentV - 0.490)/(0.550-0.490) // points at 0.51 and 0.53 are in line
  else if PresentV < 0.565 then local_TCycles = 12 + (15-12)*(PresentV - 0.550)/(0.565-0.550)
  else
    local_TCycles = 1E6 // shouldn't happen, but basically infinite time
  end
  if not StallVoltTimeSet Then Begin
    StallVoltTimeSet = True
    // write message indicating the stall time has picked up
  end
end
else begin
  // Take the average of 1/Tcycles and 1/LastTCycles from previous time-step here.
  // This is equivalent of assuming that the voltage varied linearly between the two time-steps
  StallState = StallState + (PresentTime - LastStallVoltTime) * NominalFrequency * (0.5/local_TCycles + 0.5/LastTCycles)
  if StallState >= 1.0 then begin
    StallState = 1.0 // clip this at 1.0 for reporting purposes, no reason to show values above 1.0
    // write message indicating that we have stalled the motors
    Transition Motors to the Stall State
  end
end
end
LastStallVoltTime = PresentTime // update time
LastTCycles = local_TCycles
end
```

V terminal (pu voltage)	Tstall (cycles)
0.00	2
0.45	2
0.49	3
0.51	6
0.53	9
0.55	12
0.565	15
V>0.565	No stall (9999)