

Relay Model ATRRELAY

The following documentation was created by Eric Bahr at NorthWestern Energy in June 2015 and represents in detail what is implemented in the ATRRelay model

Added LeaveOneOn parameter on 4/1/2020



ACCELERATION TREND RELAY MODEL SPECIFICATION

*Prepared By:
Eric Bahr, PE
NorthWestern Energy*

06/08/2015



Table of Contents

1. Introduction	3
2. Purpose and Scope.....	3
3. Requirements.....	3
4. Model Specification	4
4.1. Overall Block Diagram	4
4.2. Individual Unit Calculations	5
4.3. Center of Mass	7
4.4. Look-Ahead Angle	7
4.5. Trip Logic.....	8
4.6. Trip Arbitration	21
4.7. Relaying.....	22
4.8. Input Parameters and Output Channels	23
5. Validation	24
6. Conclusion.....	24
Appendix A Trip Arbitration Example	25
Appendix B Trip List Sorting Logic	29
Appendix C Validation Output.....	31
Appendix D PowerWorld Pseudo Code	69



1. Introduction

The Acceleration Trend Relay (ATR) is a Special Protection Scheme designed to protect the customers of NorthWestern Energy from service interruptions and to prevent unstable events from disrupting the interconnected grid in Montana and Wyoming when faults and line outages occur on the Montana 500 kV transmission system.

The device consists of a computer based relay which monitors the real-time speed, acceleration, and angle of the four Colstrip units. It uses this information to assess the dynamic performance of the system and is able to detect unstable events in progress. The ATR will trip generators as necessary to restore the system to a stable operating condition when instability occurs.

2. Purpose and Scope

This document is meant to serve as a specification for a simplified model of the ATR to be used by other members of the WECC. This model is to be approved by the WECC MVWG and implemented by the software vendors for use in positive-sequence transient stability simulations. Due to the nature of the ATR, this is a dynamics only model specification and a power flow representation is not available.

3. Requirements

The following requirements apply to the ATR model and define the intended use and limitations of the model:

- The model shall provide a relatively good representation of the Acceleration Trend Relay for use in most planning and operational studies, but may not be adequate for more in-depth studies inside, or near NorthWestern Energy's system.
- The model shall work in simulations with a time step ranging from 1 to 10 milliseconds.
- The model should report to the user whenever a trip decision is made.



4. Model Specification

4.1. Overall Block Diagram

The overall block diagram for the ATR is given in Figure 1. First, individual unit speed, acceleration, and angle are calculated from the machines electrical power output (MW) and speed deviation (pu). The values from each of the four machines are then passed into a center of mass calculation where the weighted averages of speed, acceleration, and angle of the entire plant are calculated. These values are then used to calculate a projected look-ahead angle. This value, along with the individual and center of mass values are then sent to the trip logic. The trip logic calculates the amount of generation necessary to maintain stability. A trip percentage is then issued and sent to trip arbitration logic which decides which units to trip to satisfy the trip percentage. Unit trips will then be made by sending a trip signal to the individual unit relays. A detailed description is given for each portion of the model in the following sections.

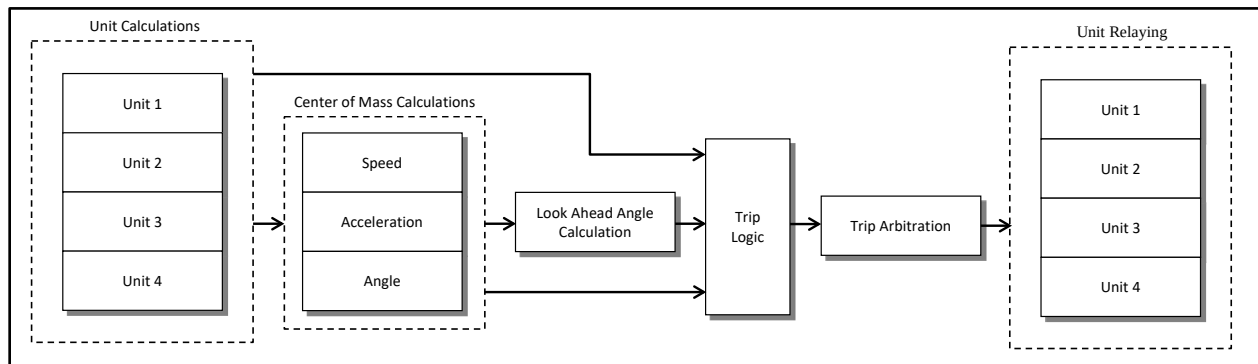


Figure 1: Overall Block Diagram



4.2. Individual Unit Calculations

The ATR uses three primary units of measurement for its calculations: speed, angle, and acceleration.

Speed

The speed value represents a weighted sum for the measured tooth-wheel counts at both ends of the rotor with a low-pass filter applied. A speed deviation value is then calculated by comparing present speed against a long-term average. One unit of speed corresponds to $3.766\text{E-}3$ radians per second, or one thousandth of one percent of the nominal speed of a machine.

Angle

The angle value is the numerical integral of the speed value. In order to prevent the angle from increasing without bound, a long-term washout filter is applied. When units are operating at nominal speed, the angle tends to go to zero. One thousand angle counts correspond to 1.44 degrees.

Acceleration

The accelerating power is derived from subtracting the electrical power output from the mechanical power output. The electrical power output is provided by watt/var transducer measurements and a first approximation of mechanical power is derived from a low pass filter applied to the watt measurement. The mechanical power estimate is adjusted so that the acceleration calculated as the difference of electrical and mechanical power matches a derivative of speed. One unit of acceleration corresponds to 1 MW of accelerating power on a small unit and 2.75 MW of accelerating power on a large unit.

Figure 2 shows a good approximation of how these three values can be calculated for each unit from within the dynamic simulation software. The model is to be applied to each generator at Colstrip where $P_{gen(MW)}$ is the electrical power output of the generator in megawatts and ω is the generator speed deviation in per unit. These values are to be taken from the internal models already applied to each unit. The gains for each unit are given in Table 1.

Unit	Acceleration Gain	Speed Gain
1	1.00	12.51
2	1.00	12.51
3	1/2.75	34.43
4	1/2.75	34.43

Table 1: Individual Unit Gains

The outputs for each of the unit calculations are $uXspd$, $uXang$, and $uXacl$ where X is a number, 1 through 4, representing which unit it corresponds to.

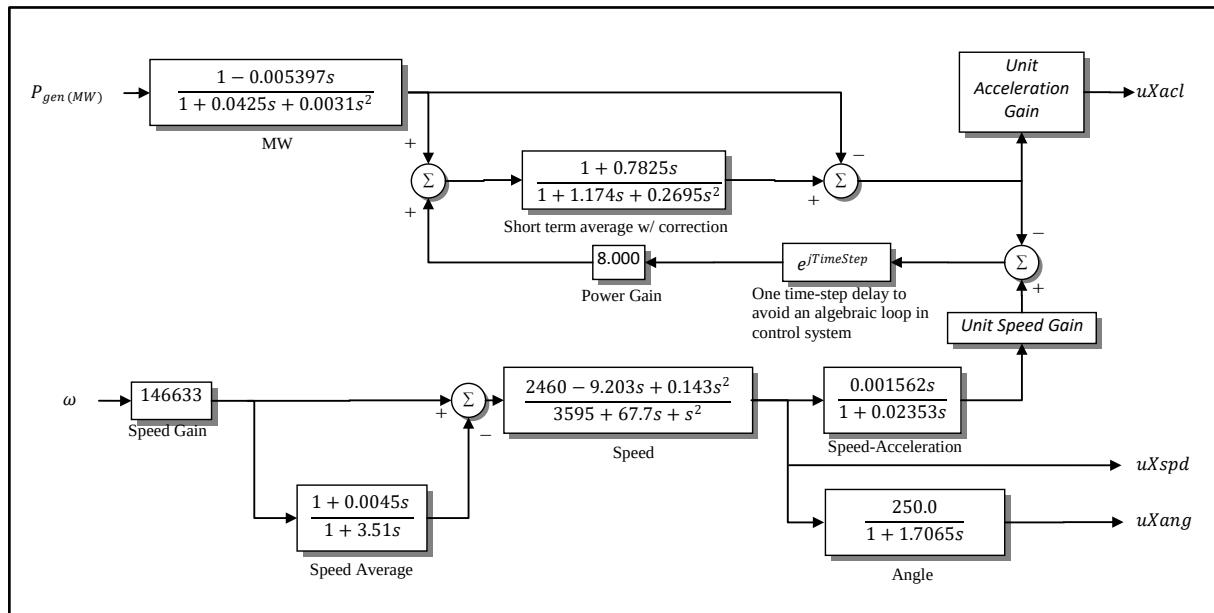


Figure 2: Individual Unit Calculations - Detailed

The filtering on the ATR is designed to filter out the torsional oscillations seen on each end of the turbine shaft by the tooth-wheel sensors. Since the spring-mass model of the generator is generally not modeled in most transient simulations, these oscillations are not present and many of the higher frequency filtering is not needed. The time-constants of some of these filters are also very close to common sampling frequencies used in these simulations, therefore a simplified block diagram was derived. This model is shown in Figure 3 and still provides a close approximation to the more detailed model.

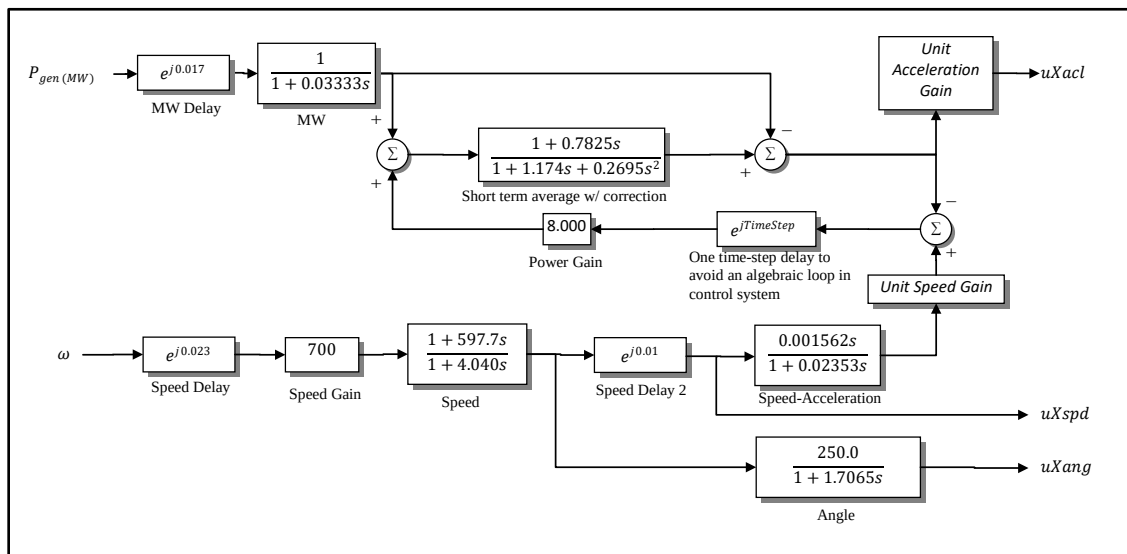


Figure 3: Individual Unit Calculations - Simplified



4.3. Center of Mass

The center of mass calculation is simply the appropriately weighted average of speed, angle, or acceleration of each online unit. The same calculation is used to find the center of mass speed, center of mass angle, and center of mass acceleration. A very simplified center of mass calculation is given by the following equation:

$$cmval = \frac{u1stat * u1val + u2stat * u2val + u3stat * 2.75 * u3val + u4stat * 2.75 * u4val}{u1stat + u2stat + u3stat * 2.75 + u4stat * 2.75}$$

where $uXstat$ is the unit online status with online status (online=1, offline=0) and val is the *spd*, *ang*, or *acl*. Appropriate measures would have to be taken when all 4 units are offline to ensure there are no divide by zero issues. The moment of inertia of units 3 and 4 is approximately 2.75 times the moment of inertia of units 1 and 2. (Number was chosen to accommodate integer arithmetic in original implementation.)

4.4. Look-Ahead Angle

The look-ahead angle is a projected value of angle 6 cycles into the future using a constant acceleration assumption. The units of look-ahead angle are the same as that of angle. The center of mass look-ahead angle is given by the following equation:

$$cmla6ang = cmang + 15.0 * cmspd + 35.971 * cmacl$$

where $cmang$ is the center of mass angle, $cmspd$ is the center of mass speed, and $cmacl$ is the center of mass acceleration.



4.5. Trip Logic

The ATR has 11 different algorithms it uses to detect instability; 10 of them which use the center of mass data and 1 that uses individual unit data. An overview of the trip logic is shown in Figure 4. A detailed description of each block is provided in this section. When an algorithm detects an event which meets the tripping criterions, it issues a trip call in percent. Trip algorithms are run in the order specified at each time step. A trip request is issued immediately when a trip algorithm determines that a trip is required. Except for TDAC, Dynamic Oscillation, 2nd Peak Over-speed, and ANAC, a trip algorithm that issues a trip request cannot issue another trip request until the Gross Enable is reset. ANAC is blocked after making a trip evaluation, but is reset by either Gross Enable being reset or a new maximum in acceleration occurring. 2nd Peak Over-speed is blocked after issuing a trip request and reset when speed becomes less than or equal to zero. If Gross Enable goes false, any remaining TDAC evaluations are canceled. The output of each algorithm will always be zero (*trippct* = 0.0) whenever the logic for the algorithm is False (0). The output of each algorithm should also be set to zero when the particular algorithm has been disabled due to a trip call. All algorithms, except the Individual Unit Over-speed, are inactive until the Gross Enable flag (*enable*) is set.

The Individual Unit Over-speed trip evaluation is made each time step after each unit's filtered speed is calculated and before any center of mass trip evaluations are made.

The center of mass trip algorithms are to be run in the following order each time step:

1. Dynamic Oscillation
2. 2nd Peak Over-speed
3. TDAC
4. SPAN
5. ANAC
6. OACL
7. ACSP
8. OSPD
9. ACAN
10. OANG

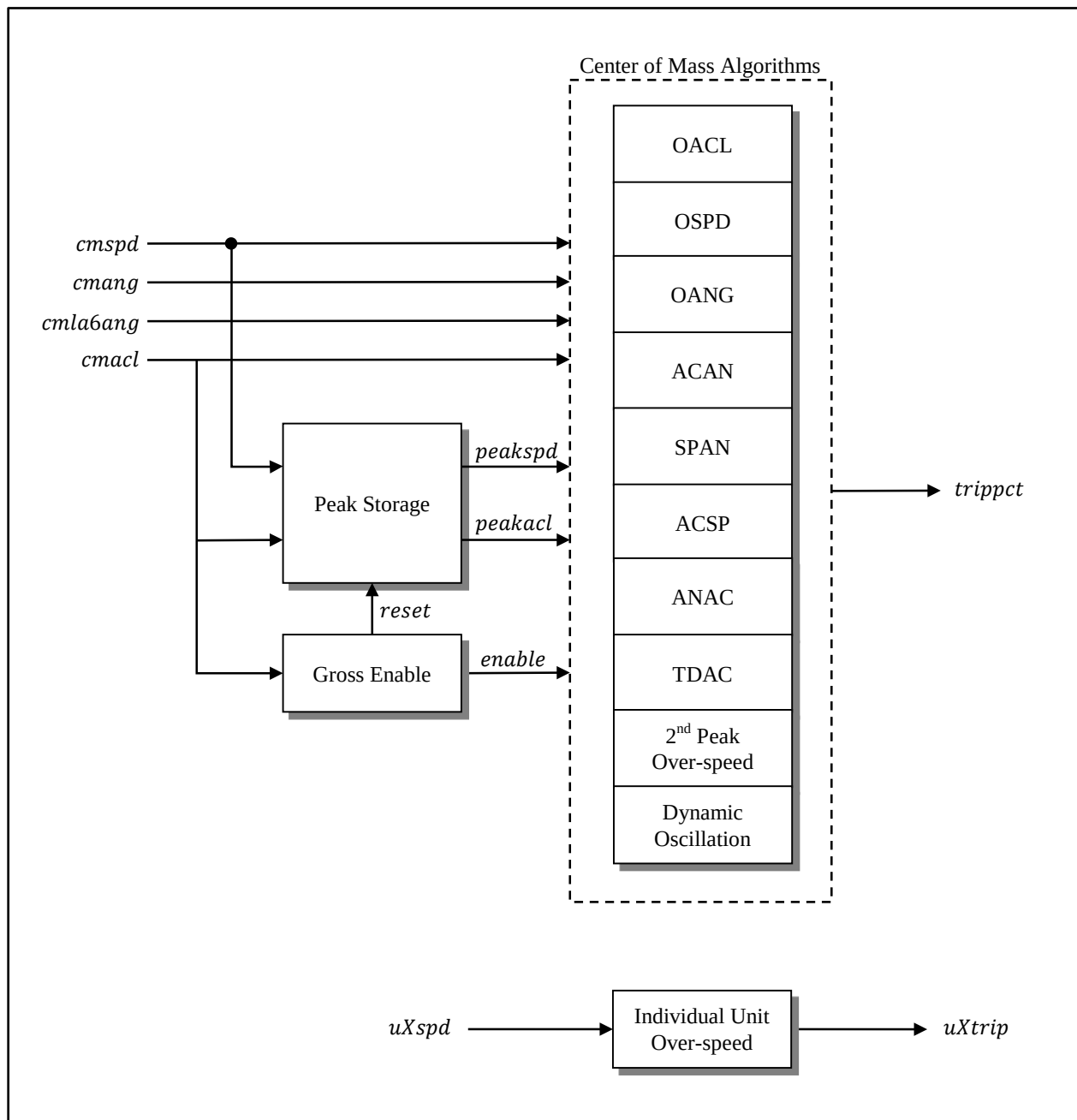


Figure 4: Trip Logic



Gross Enable

In order to prevent the ATR from issuing trips for events initiated in remote systems which cause islanding and subsequent system frequency excursions which persist, an *enable* flag is set when acceleration is greater than a threshold based on the smallest acceleration pulse for which studies have shown that unit tripping may be required. The *enable* flag remains set until speed and angle are both negative by margins greater than nominal noise values, or until an indefinite time interval of more than two seconds and less than three seconds has expired, whichever comes first. For dynamic simulation purposes, a time interval of 2.5 seconds can be used. All algorithms, except the Individual Unit Over-speed algorithm, are inactive until the *enable* flag is set. The *enable* flag is set by the following equation:

$$IF(cmacl \geq 20) THEN enable = True(1)$$

Once the *enable* flag is set, a timer is to be set for 2.5 seconds. Once the timer expires, the *enable* flag is reset.

The *enable* flag will also reset when both the following conditions are true AND speed and angle are decreasing:

Initialize:

```
speed_up = False;  
speed_reset = False;  
angle_up = False;  
angle_reset = False;  
angle_reset = false;  
speed_reset = false;
```

Determine if angle is above noise threshold:

```
If (fcmang > 1500) then begin  
    angle_up = True;  
    angle_reset = False;  
end;
```

Determine if angle below noise threshold and decreasing:

```
If (fcmang < -1500) and (angle_up) then begin
```



angle_reset = True;

end;

Determine if speed is above noise threshold:

If (fcmspd > 50) then begin

speed_up = True;

speed_reset = False;

end;

Determine if speed below noise threshold and decreasing:

If (fcmspd < -50) and (speed_up) then begin

speed_reset = True;

end;

If (angle_reset) and (speed_reset) then enable = False

If the final condition is met, both the timer and *enable* flag are reset and the algorithms become disabled. When the *enable* flag is reset, a *reset* signal is to be sent to the Peak Storage logic to reset the peak values.

Peak Storage

The peak to date values of *cmspd* and *cmacr* are stored as values *peakspd* and *peakacr*. These values are updated each time step if the current value of *cmspd* or *cmacr* exceed the stored values of *peakspd* or *peakacr*. These values are reset when the *reset* flag is sent from the Gross Enable logic.

Algorithms

In order for the algorithm to be functional Enable needs to be TRUE.

Thus If Enable = TRUE then:

Dynamic Oscillation

When dynamic oscillations become larger than the normal noise, positive and negative values are subtracted to compute the peak-to-peak amplitude and successive values of the amplitude are compared.

A peak is not considered a peak unless it has reached a value of ± 100 counts. The peak is also not detected until it has gone up or down by either 20 percent of the peak value, or 12 counts, whichever is greater. See **Error! Reference source not found.** for a representation of how this algorithm may behave.



Thus:

If Peak High >100 then begin, or (For peak High detection)

If Peak Low <100 then begin (For Peak Low detection)

Peak Detected: if Peak High > 0.20 or (Peak High - cmspd)> 12 counts then (if going down)

if Peak Low > 0.20 or abs(Peak Low - cmspd) > 12 counts then begin (if going up)

Amplitude = Peak High – Peak Low

The following must all be true for algorithm to issue a trip:

- *Peak-to-peak amplitude must increase for the last 3 consecutive cycles*
- *The amplitude must be greater than 200 counts*
- *The previous minimum peak must be less than -165 counts*
- *The consecutive peaks must have occurred within the last 2 seconds from each other*

If all are true, then trippct = 0.14..

End

End

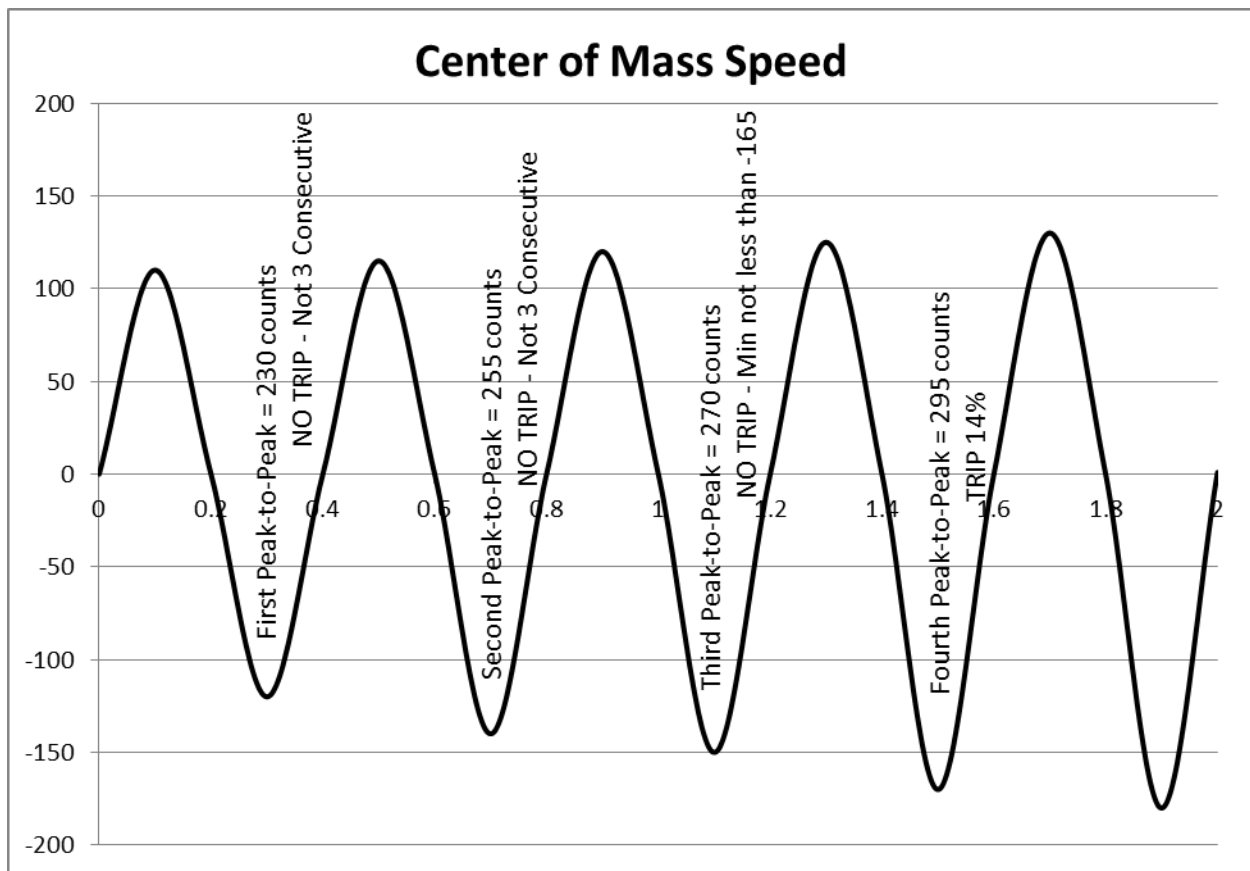


Figure 5: Dynamic Oscillation



Second Peak Over-speed

When a positive local maximum speed greater than normal noise is detected, the value is stored. Until the speed goes negative, subsequent values of speed are compared to the previous peak. If a new value exceeds the old peak, this algorithm calls for a trip. The amount to be tripped depends on the magnitude of the first peak. If the first peak is 450 counts or greater, the trip amount is 81 percent. If the first peak is 175 or greater, but less than 450, the trip amount is 48 percent. If the initial peak is less than 175, the algorithm will not issue a trip.

A peak is not considered a peak unless it has reached a value of 100 counts. The peak is also not detected until it has gone down by either 20 percent of the peak value, or 12 counts, whichever is greater. See Figure 6 for a representation of how this algorithm behaves.

Thus:

Initialize Peak Old = 0;

Peak Detected:

if Peak High > 0.20 or (Peak High - cmspd)> 12 counts then begin

If Peak High > Peak Old then begin

If (Peak Old > 450) then trippct = 0.81;

If (Peak Old <= 450) and (Peak Old > 175) then trippct = 0.48;

Peak Old = Peak High

end

If cmspd < 0 then Peak Old = 0;

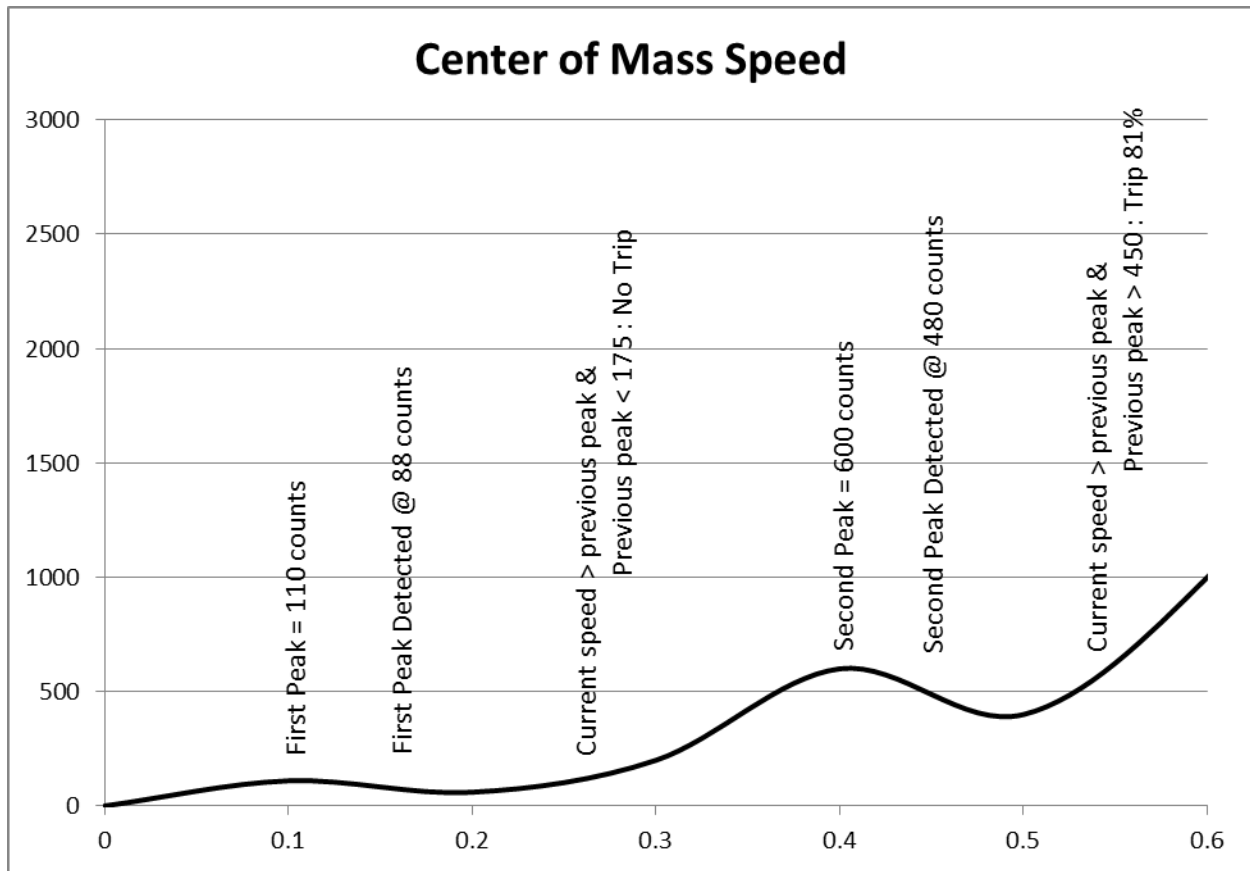


Figure 6: Second Peak Over-speed



Time Triggered Acceleration (TDAC)

When acceleration exceeds a threshold of 20 counts and acceleration has increased by at least 10 counts in the last 4 electrical cycles, a timer (*tdactimer*) is initiated. The peak-to-date acceleration is then used to compute an acceleration threshold at various time intervals. When the timer reaches a time interval limit, the acceleration and peak acceleration values are compared to threshold equations. If the thresholds are exceeded, and the angle is greater than 4000 counts, a trip is issued. There are six different time intervals, each with varying levels of tripping thresholds. Once the time expires and the algorithm goes through the calculations and decides whether or not to trip, that TDAC algorithm is disabled and no longer able to make a trip decision. The time delays range from 5.2 cycles to 12 cycles. The following describes the logic for each time step.

Save values of *cmacl* for the previous 4 cycles (*Old_cmacl*)

Initialize:

```
BlockedTDAC10 := false;  
BlockedTDAC8 := false;  
BlockedTDAC68 := false;  
BlockedTDAC6 := false;  
BlockedTDAC52 := false;  
tdacDoAlgorithms := false  
tdactimerSet := false  
tdactimerInitialized := false
```

If cmacl > 20 then begin

If (cmacl – Old_cmacl) at any time in the previous 4 cycles then begin

tdacDoAlgorithms := *true*

tdactimerSet := *true*

end

else begin

tdactimerSet := *false*

tdactimerInitialized := *false*



end

If tdacDoAlgorithms then begin

If tdactimerSet and (not tdactimerInitialized) then begin

tdactimerInitial := CurrentTime

tdactimerInitialized := true;

end

tdactimer := CurrentTime – tdactimerInitial

TDAC 5.2

IF(tdactimer $\geq$ 0.0867 sec) and (not BlockedTDAC10)THEN

IF((peakacl $\geq$ 25) and (cmang > 4000)) THEN

*IF(cmacl $\geq$ 115 + 0.225 * peakacl) THEN trippct = 0.81*

*ELSEIF(cmacl $\geq$ 25 + 0.714 * peakacl) THEN trippct = 0.81*

ELSEIF(cmacl $\geq$ 148) THEN trippct = 0.81

BlockedTDAC52:= true

TDAC 6.0

IF(tdactimer $\geq$ 0.1000 sec) and (not BlockedTDAC6)THEN

IF((peakacl $\geq$ 75) and (cmang > 4000)) THEN

*IF(cmacl $\geq$ 50 + 0.450 * peakacl) THEN trippct = 0.81*

*ELSEIF(cmacl $\geq$ 60 + 0.300 * peakacl) THEN trippct = 0.64*

ELSEIF(cmacl $\geq$ 103) THEN trippct = 0.64

*ELSEIF(cmacl $\geq$ 40.8 + 0.256 * peakacl) THEN trippct = 0.48*

ELSEIF(cmacl $\geq$ 85) THEN trippct = 0.48

BlockedTDAC6 := true

TDAC 6.8

IF(tdactimer $\geq$ 0.1133 sec) and (not BlockedTDAC68) THEN

IF((peakacl $\geq$ 75) and (cmang > 4000)) THEN

*IF(cmacl $\geq$ 80 + 0.150 * peakacl) THEN trippct = 0.81*



ELSEIF(cmacl $\geq$ 83) THEN trippct = 0.64

ELSEIF(cmacl $\geq$ 53) THEN trippct = 0.48

BlockedTDAC68 := true

TDAC 8.0

IF(tdactimer $\geq$ 0.1333 sec) and (not BlockedTDAC8) THEN

IF((peakacl $\geq$ 25) and (cmang > 4000)) THEN

*IF(cmacl $\geq$ 80 – 0.300 * peakacl) THEN trippct = 0.48*

*ELSEIF(cmacl $\geq$ 25 – 0.050 * peakacl) THEN trippct = 0.31*

BlockedTDAC8 := true

TDAC 10.0

IF(tdactimer $\geq$ 0.1667 sec) and (not BlockedTDAC10) THEN

IF((peakacl $\geq$ 25) and (cmang > 4000)) THEN

*IF(cmacl $\geq$ 80 – 0.525 * peakacl) THEN trippct = 0.48*

*ELSEIF(cmacl $\geq$ 50 – 0.500 * peakacl) THEN trippct = 0.31*

ELSEIF(cmacl $\geq$ 13) THEN trippct = 0.31

BlockedTDAC10 := true

TDAC 12.0

IF(tdactimer $\geq$ 0.2000 sec) THEN

IF((peakacl $\geq$ 25) and (cmang > 4000)) THEN

*IF(cmacl $\geq$ 70 – 0.500 * peakacl) THEN trippct = 0.48*

*ELSEIF(cmacl $\geq$ 55 – 0.583 * peakacl) THEN trippct = 0.31*

ELSEIF(cmacl $\geq$ 10) THEN trippct = 0.31

tdacDoAlgorithms := false

end

Speed vs Look-Ahead Angle (SPAN)

When speed becomes larger than normal noise, the peak-to-date value of speed is used to compute the largest allowable value of look-ahead angle. The following linear equation will trip 14 percent when true:



$$IF(cmla6ang \geq 12000 + 32.5 * peakspd) THEN trippct = 0.14$$

Angle Triggered Acceleration (ANAC)

The peak-to-date acceleration is used to compute six acceleration thresholds when the angle reaches 16000 counts. When this occurs, the acceleration at angle equal to 16000 is compared to each computed threshold in descending order beginning with the highest percentage. The algorithm will then issue a trip of the highest percentage. Therefore, the ATR must find the following equation to be true in order to compute the thresholds:

$$IF(cmang \geq 16000) THEN anac_enable = True (1)$$

Once *anac_enable* is found to be True (1), then the following equations will be calculated immediately and will determine how much to trip:

$$IF(cmacl \geq 215 - 0.75 * peakacl) THEN trippct = 0.81$$

$$ELSEIF(cmacl \geq 46) THEN trippct = 0.81$$

$$ELSEIF(cmacl \geq 75.33 - 0.3167 * peakacl) THEN trippct = 0.64$$

$$ELSEIF(cmacl \geq -4) THEN trippct = 0.48$$

$$ELSEIF(cmacl \geq 20 - 0.30 * peakacl) THEN trippct = 0.48$$

$$ELSEIF(cmacl \geq 32 - 0.45 * peakacl) THEN trippct = 0.31$$

Over-Acceleration (OACL)

The Over-Acceleration (OACL) algorithm issues an 81 percent trip when both the acceleration and angle simultaneously exceed thresholds given by the following equation:

$$IF((cmacl \geq 190) AND (cmang \geq 4000)) THEN trippct = 0.81$$

Acceleration vs Speed (ACSP)

When acceleration becomes larger than normal noise, the peak-to-date value of acceleration is used to compute the largest allowable value of speed. The following linear equation will trip 48 percent when true:

$$IF(cmspd \geq 170 + 3.6 * peakacl) AND (cmang > 4000) THEN trippct = 0.48$$

Over-Speed (OSPD)

The Over-Speed (OSPD) algorithm issues a 48 percent trip when speed reaches the threshold given in the following equation:

$$IF(cmspd \geq 840) THEN trippct = 0.48$$



Acceleration vs Look-Ahead Angle (ACAN)

When acceleration becomes larger than normal noise, the peak-to-date value of acceleration is used to compute the largest allowable value of look-ahead angle. The following linear equation will trip 14 percent when true:

$$IF(cmla6ang \geq 12000 + 162.5 * peakacl) THEN trippct = 0.14$$

Over-Angle (OANG)

The Over-Angle (OANG) algorithm issues a 14 percent trip when the look-ahead angle reaches the threshold given in the following equation:

$$IF(cmla6ang \geq 25000) THEN trippct = 0.14$$

Individual Unit Over-speed

Each individual unit may be tripped if the speed of that unit exceeds a threshold given by the following equation:

$$IF(uXspd \geq 1170) THEN uXtrip = True(1)$$



4.6. Trip Arbitration

The trip arbitration process is responsible for deciding which units to trip based on a percent trip (*trippct*) input. The initialization and runtime processes are listed below with a detailed example in Appendix A.

Initialization

When the ATR model initializes, the Trip List is generated. The Trip List consists of all possible tripping combinations at Colstrip with 16 possible combinations ranging from zero units, to all four units. The MW output of each unit (*u1mw*, *u2mw*, *u3mw*, and *u4mw*), total plant output (*totalmw*), and the total MW output of each combination (*combomw*) are then stored. An example of what the trip combination structure might look like is as follows:

$$tripcombo[x] = (u1trip, u2trip, u3trip, u4trip, combomw)$$

Any easy way to think of the trip list is to consider it as a 4-bit binary truth table, where each bit represents a unit to be tripped.

The trip combinations are then sorted by the preference switches specified. The possible preference switches are as follows:

- Unit 1 vs Unit 2
- Unit 3 vs Unit 4
- Small Unit vs Large Unit
- Large Unit vs Two Small Units

For example; if Unit 1 is chosen over Unit 2, Unit 1 will always be chosen to trip first and appear higher in the list. If a large unit is chosen over two small units, the ATR will try to trip only a large unit to satisfy the required *trippct*. Units 1 and 2 are considered small units, while units 3 and 4 are considered large units. When deciding between two combinations with equal preference, such as one large and one small each with one preferred unit and one non-preferred unit, the combination with the lowest *combomw* value will rank higher in the list (i.e. trip first). Code for how the trip list is sorted can be seen in Appendix B.

Runtime

Once a positive, non-zero value of *trippct* is issued, that value is passed to the Trip Arbitration runtime logic which is responsible for deciding what units to trip. The *totalmw* value is then updated based on the total plant output that is currently online and available for tripping. If a unit has tripped within the last half second (or 2 seconds if trip request came from the Dynamic Oscillation algorithm) the MW value (*prevtripmw*) is included in the *totalmw* amount. The preceding unit trip may have come from any source, including plant relaying. A unit is considered tripped when the ATR makes a decision to trip a unit or the breaker for the unit opens. If the trip request comes from the Dynamic Oscillation algorithm, any unit trip within the preceding 2 seconds is subtracted from the trip request amount.


$$prevtripmw = [\text{MW tripped within last .5 second or 2 seconds if from Dynamic Oscillation}]$$
$$totalmw = [\text{MW currently online}] + prevtripmw$$

The amount of generation to trip is then calculated by the following equation:

$$tripreqmw = trippct * totalmw$$

If there has been a unit trip within the last half second, then the tripped generation amount will be calculated and subtracted out of the requested trip amount.

$$tripmw = tripreqmw - prevtripmw$$

The ATR will then go through the Trip List and remove any combinations that have the tripped or offline units. Following this process, the ATR will then go through each combination in the Trip List until the following equation is satisfied.

$$combomw \geq tripreqmw$$

The ATR will then issue a trip to the units specified by the *u1trip*, *u2trip*, *u3trip*, and *u4trip* values in the chosen Trip List combination.

4.7. Relaying

Unit Tripping

The individual unit relaying is responsible for tripping the breaker to the specified unit. Once *u1trip*, *u2trip*, *u3trip*, and *u4trip* are set to True (1), a delay timer will start. The delay is defined by the input variables *u1delay*, *u2delay*, *u3delay*, and *u4delay*. Once the timer expires, the breaker to that unit will open and trip the unit offline. Once a unit trips offline, the time it occurred should be stored internally to be used by Trip Arbitration.

Load Tripping

When a unit trips offline, some of the station service load also drops off after a certain period of time. The time period is specified by the input variable *auxdelay*. Once the ATR model sees a unit trip offline, it should initiate a timer with a delay specified by the input variable *auxdelay*. Once the timer expires, the ATR should trip 20% of the load at that bus. Note that the actual ATR does not perform this role as it is more a part of the orderly shutdown of the plant.



4.8. Input Parameters and Output Channels

The following are the required inputs that are to be specified by the user.

Name	Description	Typical Value
<i>u1busnum</i>	Unit 1 bus number	-
<i>u1machid</i>	Unit 1 machine ID	-
<i>u2busnum</i>	Unit 2 bus number	-
<i>u2machid</i>	Unit 2 machine ID	-
<i>u3busnum</i>	Unit 3 bus number	-
<i>u3machid</i>	Unit 3 machine ID	-
<i>u4busnum</i>	Unit 4 bus number	-
<i>u4machid</i>	Unit 4 machine ID	-
<i>u1delay</i>	Unit 1 relay and breaker time delay (sec)	0.0500
<i>u2delay</i>	Unit 2 relay and breaker time delay (sec)	0.0500
<i>u3delay</i>	Unit 3 relay and breaker time delay (sec)	0.0667
<i>u4delay</i>	Unit 4 relay and breaker time delay (sec)	0.0667
<i>auxdelay</i>	Auxiliary load trip time delay (sec)	0.1427
<i>u1vsu2</i>	Unit 1 (0) vs unit 2 (1) preference switch selection	0
<i>u3vsu4</i>	Unit 3 (0) vs unit 4 (1) preference switch selection	0
<i>smvslg</i>	Small unit (0) vs large unit (1) preference switch selection	0
<i>lgvs2sm</i>	Large unit (0) vs 2 small unit (1) preference switch selection	0

Table 2: Input Variables

Name	Description
<i>u1spd</i>	Unit 1 filtered speed
<i>u1ang</i>	Unit 1 filtered angle
<i>u1acl</i>	Unit 1 filtered acceleration
<i>u2spd</i>	Unit 2 filtered speed
<i>u2ang</i>	Unit 2 filtered angle
<i>u2acl</i>	Unit 2 filtered acceleration
<i>u3spd</i>	Unit 3 filtered speed
<i>u3ang</i>	Unit 3 filtered angle
<i>u3acl</i>	Unit 3 filtered acceleration
<i>u4spd</i>	Unit 4 filtered speed
<i>u4ang</i>	Unit 4 filtered angle
<i>u4acl</i>	Unit 4 filtered acceleration
<i>cmspd</i>	Center of mass speed
<i>cmang</i>	Center of mass angle
<i>cmacl</i>	Center of mass acceleration
<i>cmla6ang</i>	Center of mass look ahead angle
<i>peakspd</i>	Peak to date center of mass speed
<i>peakacl</i>	Peak to date center of mass acceleration
<i>enable</i>	Gross Enable flag (enabled (1) and disabled (0))
<i>trippct</i>	Requested trip percent

Table 3: Output Channels



5. Validation

The model presented in this document was modeled in both MATLAB Simulink and PowerWorld and compared against the output of the actual ATR source code. Two GE format WECC base cases were used in testing; 15hs4a1e and 15ls1a1e. A variety of 500 kV contingencies known to cause ATR action were simulated for each base case. The outputs of each ATR model were then compared for accuracy. Detailed data output of each run can be seen in Appendix C.

The results of the validation runs show that the simplified model provides a very similar output to that of the actual ATR source code. While there are, at times, differences in trip times and percentages, each run has been analyzed and the trip times and percentages have been deemed valid regardless of these differences. The results demonstrate that this simplified model will trip when necessary, and will not trip when unnecessary.

It should also be noted that for each of the cases that showed different trip percentages between ATR models, the acceleration values used in the TDAC algorithm that called for the trip were such that they were right on the edge of the next trip threshold. Therefore any slight change in system conditions or model variations could cause it to push over the edge onto the next higher trip percentage. This variability is perfectly acceptable and common in the actual ATR during real world events.

The results of the validation also show that the model provided by PowerWorld matches the MATLAB model near exactly and that the model provided in this specification can be modeled and utilized in common transient stability tools

6. Conclusion

This model specification is a simplified representation of the Acceleration Trend Relay. This is to be used for simulating an approximate response of the ATR for the purposes of operations and planning studies. The model shows a close approximation to that of the actual ATR and NorthWestern Energy approves it for use by other entities of the WECC.

7. Model Variations

In December 2016 a new parameter named “GenNum” was added to the ATRRELAY model. If GenNum = 2 then generator to which model is assigned is unit 2 and then Gen2 would represent unit 1, and so on. Also NOTE that the results of the gens are now going to be in a different order if the GenNum is not 1.

In April 2020 a new parameter named “LeaveOneOn” was added to the ATRRELAY model. Set the LeaveOneOn parameter to a non-zero value to force at least one generator to remain online when a tripping command is sent. The generator with the smallest initial MW output will remain on.



Appendix A Trip Arbitration Example

Initialization

Consider the Colstrip output shown in Table 4.

	Output (MW)
<i>u1mw</i>	200
<i>u2mw</i>	300
<i>u3mw</i>	700
<i>u4mw</i>	800
<i>totalmw</i>	2000

Table 4: Trip Arbitration Example - Unit Output

In this scenario, the starting Trip List would initialize to the following:

Combo #	<i>u1trip</i>	<i>u2trip</i>	<i>u3trip</i>	<i>u4trip</i>	<i>combomw</i>
0	0	0	0	0	0
1	1	0	0	0	200
2	0	1	0	0	300
3	1	1	0	0	500
4	0	0	1	0	700
5	1	0	1	0	900
6	0	1	1	0	1000
7	1	1	1	0	1200
8	0	0	0	1	800
9	1	0	0	1	1000
10	0	1	0	1	1100
11	1	1	0	1	1300
12	0	0	1	1	1500
13	1	0	1	1	1700
14	0	1	1	1	1800
15	1	1	1	1	2000

Table 5: Trip Arbitration Example - Trip List



The trip combination list would then be sorted by the preference switches. An example of the preference switches is provided in Table 6 where it is set such that Unit 2 would trip before Unit 1, Unit 4 would trip before Unit 3, a small unit would trip before a large, and a large would trip before 2 small units.

Unit 1 vs Unit 2	Unit 2
Unit 3 vs Unit 4	Unit 4
Small vs Large	Small
1 Large vs 2 Small	1 Large

Table 6: Trip Arbitration Example - Preference Switches

The previous Combination List would then be sorted to what is shown in Table 7.

Combo #	<i>u1trip</i>	<i>u2trip</i>	<i>u3trip</i>	<i>u4trip</i>	<i>combomw</i>
0	0	0	0	0	0
2	0	1	0	0	300
1	1	0	0	0	200
8	0	0	0	1	800
4	0	0	1	0	700
3	1	1	0	0	500
10	0	1	0	1	1100
6	0	1	1	0	1000
9	1	0	0	1	1000
5	1	0	1	0	900
12	0	0	1	1	1500
11	1	1	0	1	1300
7	1	1	1	0	1200
14	0	1	1	1	1800
13	1	0	1	1	1700
15	1	1	1	1	2000

Table 7: Trip Arbitration Example - Trip List (Sorted)



Runtime

Consider the following ATR trip calls which may be similar to what is seen during a 500kV fault with an unsuccessful reclose:

Trip Call	Time (cycles)	<i>trippct</i>
ACSP	6.8	48%
ANAC	7.0	31%
TDAC 6.8	65.0	48%

Table 8: Trip Arbitration Example - Trip List

Time = 6.8 cycles

Once the first trip call is issued by the ACSP algorithm, it is passed in the Trip Arbitration logic.

The *totalmw* value is updated with the total online capacity available for tripping. Since all generation is online and there have been no trips yet, the value is calculated as follows:

$$prevtripmw = [\text{MW tripped within last half second}] = 0$$

$$totalmw = totalmw + prevtripmw = 2000$$

Next, the required trip amount would be calculated by the following equation:

$$tripreqmw = trippct * totalmw = 0.48 * 2000 = 960$$

The actual MW value to trip is then calculated. Since there have been no previous trips, it will equal the following:

$$tripmw = tripreqmw - prevtripmw = 960$$

The ATR will then go through the Trip List and remove any combinations with units that are currently offline. Since there have been no units tripped prior to the ACSP trip call, the Trip List and *tripmw* will remain the same. The ATR will then start at the top of the Trip List and work its way down until the *tripmw* amount is reached. In this case the ATR would choose to trip Combo #10 which consists of Unit 2 and Unit 4 for a total trip amount of 1100 MW. The following unit trips would then be issued:

$$u2trip = u4trip = \text{True (1)}$$

Time = 7.0 cycles When the ANAC call is received, the same process will occur and the available online generation is calculated:

$$totalmw = 200 + 700 = 900$$

Since the two tripped units were tripped within the last 30 cycles, they are added back into the total MW value:



$$prevtripmw = [\text{MW tripped within last half second}] = 1100$$

$$totalmw = totalmw + prevtripmw = 900 + 1100 = 2000$$

The required trip amount will then equal:

$$tripreqmw = 0.31 * 2000 = 620$$

Since units have tripped within the last half second, the previous trip amount will be subtracted from the required amount.

$$tripmw = tripreqmw - prevtripmw = 620 - 1100 = -480$$

The ATR will then remove any combinations from the list that include the tripped units. The Trip List will now be:

Combo #	u1trip	u2trip	u3trip	u4trip	combomw
0	0	0	0	0	0
1	1	0	0	0	200
4	0	0	1	0	700
5	1	0	1	0	900

Table 9: Trip Arbitration Example - Trip List (2nd Sort)

Since the requested *tripmw* value is negative, the ATR will not trip any units.

Time = 65.0 cycles

Next, when TDAC 6.0 call is received, the following calculations are made:

$$prevtripmw = 0$$

$$totalmw = 900 + 0 = 900$$

$$tripreqmw = 0.48 * 900 = 432$$

$$tripmw = 432 - 0 = 432$$

Since there have been no new trips since the last time the Trip List was parsed, it will remain the same as sorted earlier in Table 9: Trip Arbitration Example - Trip List (2nd Sort)Table 9. The ATR will then go through the remaining Trip List until it finds the trip amount to satisfy the *tripmw* amount. For this call, the ATR would choose Combo #4 and trip Unit 3 by issuing the following:

$$u3trip = \text{True (1)}$$



Appendix B Trip List Sorting Logic

The following Python code is an example of how the Trip List can be sorted based on Preference Switches.

```
if self.sdk.parameters['preference_switches'] == "1,3,S,2S":
    if combo14 < combo23:
        sorted_array = [0,1,2,3,4,8,5,9,6,10,12,7,11,13,14,15]
    else:
        sorted_array = [0,1,2,3,4,8,5,6,9,10,12,7,11,13,14,15]
elif self.sdk.parameters['preference_switches'] == "2,3,S,2S":
    if combo24 < combo13:
        sorted_array = [0,2,1,3,4,8,6,10,5,9,12,7,11,14,13,15]
    else:
        sorted_array = [0,2,1,3,4,8,6,5,10,9,12,7,11,14,13,15]
elif self.sdk.parameters['preference_switches'] == "1,4,S,2S":
    if combo24 > combo13:
        sorted_array = [0,1,2,3,8,4,9,5,10,6,12,11,7,13,14,15]
    else:
        sorted_array = [0,1,2,3,8,4,9,10,5,6,12,11,7,13,14,15]
elif self.sdk.parameters['preference_switches'] == "2,4,S,2S":
    if combo14 > combo23:
        sorted_array = [0,2,1,3,8,4,10,6,9,5,12,11,7,14,13,15]
    else:
        sorted_array = [0,2,1,3,8,4,10,9,6,5,12,11,7,14,13,15]
elif self.sdk.parameters['preference_switches'] == "1,3,L,2S":
    sorted_array = [0,3,4,8,1,2,12,5,6,9,10,13,14,7,11,15]
elif self.sdk.parameters['preference_switches'] == "2,3,L,2S":
    sorted_array = [0,3,4,8,2,1,12,6,5,9,10,13,14,7,11,15]
elif self.sdk.parameters['preference_switches'] == "1,4,L,2S":
    sorted_array = [0,3,8,4,1,2,12,9,10,5,6,13,14,11,7,15]
elif self.sdk.parameters['preference_switches'] == "2,4,L,2S":
    sorted_array = [0,3,8,4,2,1,12,10,9,6,5,14,13,11,7,15]
elif self.sdk.parameters['preference_switches'] == "1,3,S,L":
    if combo14 < combo23:
        sorted_array = [0,1,2,4,8,3,5,9,6,10,12,7,11,13,14,15]
    else:
        sorted_array = [0,1,2,4,8,3,5,6,9,10,12,7,11,13,14,15]
elif self.sdk.parameters['preference_switches'] == "2,3,S,L":
    if combo24 < combo13:
        sorted_array = [0,2,1,4,8,3,6,10,5,9,12,7,11,14,13,15]
    else:
        sorted_array = [0,2,1,4,8,3,6,5,10,9,12,7,11,14,13,15]
elif self.sdk.parameters['preference_switches'] == "1,4,S,L":
    if combo24 > combo13:
        sorted_array = [0,1,2,8,4,3,9,5,10,6,12,11,7,13,14,15]
    else:
        sorted_array = [0,1,2,8,4,3,9,10,5,6,12,11,7,13,14,15]
elif self.sdk.parameters['preference_switches'] == "2,4,S,L":
    if combo14 > combo23:
        sorted_array = [0,2,1,8,4,3,10,6,9,5,12,11,7,14,13,15]
    else:
        sorted_array = [0,2,1,8,4,3,10,9,6,5,12,11,7,14,13,15]
elif self.sdk.parameters['preference_switches'] == "1,3,L,L":
    if combo14 > combo23:
        sorted_array = [0,4,8,1,2,12,5,6,9,10,3,13,14,7,11,15]
    else:
        sorted_array = [0,4,8,1,2,12,5,9,6,10,3,13,14,7,11,15]
elif self.sdk.parameters['preference_switches'] == "2,3,L,L":
    if combo24 > combo13:
        sorted_array = [0,4,8,2,1,12,6,5,10,9,3,14,13,7,11,15]
    else:
        sorted_array = [0,4,8,2,1,12,6,10,5,9,3,14,13,7,11,15]
elif self.sdk.parameters['preference_switches'] == "1,4,L,L":
    if combo24 > combo13:
```



```
        sorted_array = [0,8,4,1,2,12,9,10,5,6,3,13,14,11,7,15]
    else:
        sorted_array = [0,8,4,1,2,12,9,5,10,6,3,13,14,11,7,15]
elif self.sdk.parameters['preference_switches'] == "2,4,L,L":
    if combo14 < combo23:
        sorted_array = [0,8,4,2,1,12,10,9,6,5,3,14,13,11,7,15]
    else:
        sorted_array = [0,8,4,2,1,12,10,6,9,5,3,14,13,11,7,15]
```



Appendix C Validation Output

The simplified model was modeled in both MATLAB Simulink and PowerWorld and compared against the output of the actual ATR source using the same input data. The heading of each page shows which base case and contingency the data belongs to in the format **basecase_contingency**. The following is a summary of the contingencies tested.

Label	Description
1bvgr1	Single phase fault at the Broadview end of Broadview – Garrison 500kV line 1 and successful reclosing of the line
1bvgr1_ur	Single phase fault at the Broadview end of Broadview – Garrison 500kV line 1 and unsuccessful reclosing followed by a 3 phase opening of the line.
1bvgr12	Single phase fault at the Broadview end of Broadview – Garrison 500kV lines 1&2 and successful reclosing of the lines
1bvgr12_ur	Single phase fault at the Broadview end of Broadview – Garrison 500kV lines 1&2 and unsuccessful reclosing followed by a 3 phase opening of the lines.
1csbv1	Single phase fault at the Colstrip end of Colstrip – Broadview 500kV line 1 and successful 3 phase opening of the line
1grtt12	Single phase fault at the Garrison end of Garrison – Taft 500kV lines 1&2 and successful 3 phase opening of the lines
3bvgr1	Three phase fault at the Broadview end of Broadview – Garrison 500kV line 1 and successful 3 phase opening of the line
3bvgr12	Three phase fault at the Broadview end of Broadview – Garrison 500kV lines 1&2 and successful 3 phase opening of the lines
3csbv1	Three phase fault at the Colstrip end of Colstrip – Broadview 500kV line 1 and successful 3 phase opening of the line
3csbv12	Three phase fault at the Colstrip end of Colstrip – Broadview 500kV line 1&2 and successful 3 phase opening of the lines
3grtt1	Three phase fault at the Garrison end of Garrison – Taft 500kV lines 1 and successful 3 phase opening of the line
3grtt12	Three phase fault at the Garrison end of Garrison – Taft 500kV lines 1&2 and successful 3 phase opening of the lines
3ttbb	Three phase fault at the Taft end of the Taft – Bell 500kV line and successful 3 phase opening of the line
bvgr12	No fault opening of the Broadview – Garrison 500kV lines 1&2
grtt12	No fault opening of the Garrison – Taft 500kV lines 1&2



The following code is the contingency definitions in PowerWorld .aux format:

```
DATA (TSCONTINGENCYELEMENT,  
[WhoAmI:2,TSTimeInCycles,TSTimeInSeconds,WhoAmI,TSEventString,Enabled,  
FilterName,  
    Comment,TSCTGName])  
{  
"Line GARTOW12 TO TOWNSEND 1 CKT 1"      63.0 1.050000 "Branch  
'627204' '627201' '1'" "OPEN BOTH" "CHECK" "" ""  
"1bvgr1_successful_reclose"  
"Line BROADVIEW TO TOWNSEND 1 CKT 1"     108.0 1.800000 "Branch  
'623512' '627201' '1'" "CLOSE BOTH" "CHECK" "" ""  
"1bvgr1_successful_reclose"  
"Bus BROADVIEW"      60.0 1.000000 "Bus '623512'" "FAULT SLG ADM 2.579  
-35.24" "CHECK" "" "" "1bvgr1_successful_reclose"  
"Bus BROADVIEW"      63.0 1.050000 "Bus '623512'" "CLEARFAULT" "CHECK"  
"" "" "1bvgr1_successful_reclose"  
"Line GARTOW12 TO TOWNSEND 1 CKT 1"     108.0 1.800000 "Branch  
'627204' '627201' '1'" "CLOSE BOTH" "CHECK" "" ""  
"1bvgr1_successful_reclose"  
"Line GAR1EAST TO GARTOW11 CKT 1"        63.0 1.050000 "Branch '40451'  
'627203' '1'" "OPEN BOTH" "CHECK" "" "" "1bvgr1_successful_reclose"  
"Series Cap GAR1EAST TO GARRISON CKT 1"  108.0 1.800000 "Branch  
'40451' '40459' '1'" "CLOSE BOTH" "CHECK" "" ""  
"1bvgr1_successful_reclose"  
"Line BROADVIEW TO TOWNSEND 1 CKT 1"     63.0 1.050000 "Branch  
'623512' '627201' '1'" "OPEN BOTH" "CHECK" "" ""  
"1bvgr1_successful_reclose"  
"Series Cap GAR1EAST TO GARRISON CKT 1"   63.0 1.050000 "Branch  
'40451' '40459' '1'" "OPEN BOTH" "CHECK" "" ""  
"1bvgr1_successful_reclose"  
"Line GARTOW11 TO GARTOW12 CKT 1"        63.0 1.050000 "Branch '627203'  
'627204' '1'" "OPEN BOTH" "CHECK" "" "" "1bvgr1_successful_reclose"  
"Line GARTOW11 TO GARTOW12 CKT 1"        108.0 1.800000 "Branch '627203'  
'627204' '1'" "CLOSE BOTH" "CHECK" "" "" "1bvgr1_successful_reclose"  
"Line GAR1EAST TO GARTOW11 CKT 1"        108.0 1.800000 "Branch '40451'  
'627203' '1'" "CLOSE BOTH" "CHECK" "" "" "1bvgr1_successful_reclose"  
"Line GARTOW12 TO TOWNSEND 1 CKT 1"      108.0 1.800000 "Branch  
'627204' '627201' '1'" "CLOSE BOTH" "CHECK" "" ""  
"1bvgr1_unsuccessful_reclose"  
"Bus BROADVIEW"      63.0 1.050000 "Bus '623512'" "CLEARFAULT" "CHECK"  
"" "" "1bvgr1_unsuccessful_reclose"  
"Series Cap GAR1EAST TO GARRISON CKT 1"  108.0 1.800000 "Branch  
'40451' '40459' '1'" "CLOSE BOTH" "CHECK" "" ""  
"1bvgr1_unsuccessful_reclose"  
"Line BROADVIEW TO TOWNSEND 1 CKT 1"     108.0 1.800000 "Branch  
'623512' '627201' '1'" "CLOSE BOTH" "CHECK" "" ""  
"1bvgr1_unsuccessful_reclose"
```



"Series Cap GAR1EAST TO GARRISON CKT 1" 63.0 1.050000 "Branch
'40451' '40459' '1'" "OPEN BOTH" "CHECK" "" ""
"1bvgr1_unsuccessful_reclose"
"Line BROADVIEW TO TOWNSEND 1 CKT 1" 63.0 1.050000 "Branch
'623512' '627201' '1'" "OPEN 1Phase" "CHECK" "" ""
"1bvgr1_unsuccessful_reclose"
"Line GARTOW12 TO TOWNSEND 1 CKT 1" 63.0 1.050000 "Branch
'627204' '627201' '1'" "OPEN BOTH" "CHECK" "" ""
"1bvgr1_unsuccessful_reclose"
"Series Cap GAR1EAST TO GARRISON CKT 1" 111.0 1.850000 "Branch
'40451' '40459' '1'" "OPEN BOTH" "CHECK" "" ""
"1bvgr1_unsuccessful_reclose"
"Line BROADVIEW TO TOWNSEND 1 CKT 1" 111.0 1.850000 "Branch
'623512' '627201' '1'" "OPEN BOTH" "CHECK" "" ""
"1bvgr1_unsuccessful_reclose"
"Line GARTOW11 TO GARTOW12 CKT 1" 108.0 1.800000 "Branch '627203'
'627204' '1'" "CLOSE BOTH" "CHECK" "" "" "1bvgr1_unsuccessful_reclose"
"Bus BROADVIEW" 111.0 1.850000 "Bus '623512'" "CLEARFAULT" "CHECK"
"" "" "1bvgr1_unsuccessful_reclose"
"Line GARTOW11 TO GARTOW12 CKT 1" 63.0 1.050000 "Branch '627203'
'627204' '1'" "OPEN BOTH" "CHECK" "" "" "1bvgr1_unsuccessful_reclose"
"Line GAR1EAST TO GARTOW11 CKT 1" 63.0 1.050000 "Branch '40451'
'627203' '1'" "OPEN BOTH" "CHECK" "" "" "1bvgr1_unsuccessful_reclose"
"Line GAR1EAST TO GARTOW11 CKT 1" 108.0 1.800000 "Branch '40451'
'627203' '1'" "CLOSE BOTH" "CHECK" "" "" "1bvgr1_unsuccessful_reclose"
"Bus BROADVIEW" 108.0 1.800000 "Bus '623512'" "FAULT SLG ADM 2.579
-35.24" "CHECK" "" "" "1bvgr1_unsuccessful_reclose"
"Bus BROADVIEW" 60.0 1.000000 "Bus '623512'" "FAULT SLG ADM 2.579
-35.24" "CHECK" "" "" "1bvgr1_unsuccessful_reclose"
"Bus TOWNSEND 1" 111.0 1.850000 "Bus '627201'" "OPEN" "CHECK" "" ""
"1bvgr1_unsuccessful_reclose"
"Line GAR2EAST TO GARTOW21 CKT 2" 63.0 1.050000 "Branch '40453'
'627205' '2'" "OPEN BOTH" "CHECK" "" "" "1bvgr12_successful_reclose"
"Series Cap GAR1EAST TO GARRISON CKT 1" 63.0 1.050000 "Branch
'40451' '40459' '1'" "OPEN BOTH" "CHECK" "" ""
"1bvgr12_successful_reclose"
"Bus BROADVIEW" 60.0 1.000000 "Bus '623512'" "FAULT SLG ADM 2.579
-35.24" "CHECK" "" "" "1bvgr12_successful_reclose"
"Line GARTOW11 TO GARTOW12 CKT 1" 63.0 1.050000 "Branch '627203'
'627204' '1'" "OPEN BOTH" "CHECK" "" "" "1bvgr12_successful_reclose"
"Line GAR1EAST TO GARTOW11 CKT 1" 108.0 1.800000 "Branch '40451'
'627203' '1'" "CLOSE BOTH" "CHECK" "" "" "1bvgr12_successful_reclose"
"Series Cap GAR2EAST TO GARRISON CKT 1" 63.0 1.050000 "Branch
'40453' '40459' '1'" "OPEN BOTH" "CHECK" "" ""
"1bvgr12_successful_reclose"
"Line BROADVIEW TO TOWNSEND 2 CKT 2" 108.0 1.800000 "Branch
'623512' '627202' '2'" "CLOSE BOTH" "CHECK" "" ""
"1bvgr12_successful_reclose"



```
"Line GARTOW22 TO TOWNSEND 2 CKT 2" 108.0 1.800000 "Branch
'627206' '627202' '2'" "CLOSE BOTH" "CHECK" "" ""
"1bvgr12_successful_reclose"
"Line GARTOW21 TO GARTOW22 CKT 2" 108.0 1.800000 "Branch '627205'
'627206' '2'" "CLOSE BOTH" "CHECK" "" "" "1bvgr12_successful_reclose"
"Line GAR1EAST TO GARTOW11 CKT 1" 63.0 1.050000 "Branch '40451'
'627203' '1'" "OPEN BOTH" "CHECK" "" "" "1bvgr12_successful_reclose"
"Line GARTOW12 TO TOWNSEND 1 CKT 1" 108.0 1.800000 "Branch
'627204' '627201' '1'" "CLOSE BOTH" "CHECK" "" ""
"1bvgr12_successful_reclose"
"Line GARTOW11 TO GARTOW12 CKT 1" 108.0 1.800000 "Branch '627203'
'627204' '1'" "CLOSE BOTH" "CHECK" "" "" "1bvgr12_successful_reclose"
"Line BROADVIEW TO TOWNSEND 1 CKT 1" 108.0 1.800000 "Branch
'623512' '627201' '1'" "CLOSE BOTH" "CHECK" "" ""
"1bvgr12_successful_reclose"
"Line GAR2EAST TO GARTOW21 CKT 2" 108.0 1.800000 "Branch '40453'
'627205' '2'" "CLOSE BOTH" "CHECK" "" "" "1bvgr12_successful_reclose"
"Series Cap GAR2EAST TO GARRISON CKT 1" 108.0 1.800000 "Branch
'40453' '40459' '1'" "CLOSE BOTH" "CHECK" "" ""
"1bvgr12_successful_reclose"
"Series Cap GAR1EAST TO GARRISON CKT 1" 108.0 1.800000 "Branch
'40451' '40459' '1'" "CLOSE BOTH" "CHECK" "" ""
"1bvgr12_successful_reclose"
"Bus BROADVIEW" 63.0 1.050000 "Bus '623512'" "CLEARFAULT" "CHECK"
"" "" "1bvgr12_successful_reclose"
"Line GARTOW22 TO TOWNSEND 2 CKT 2" 63.0 1.050000 "Branch
'627206' '627202' '2'" "OPEN BOTH" "CHECK" "" ""
"1bvgr12_successful_reclose"
"Line GARTOW21 TO GARTOW22 CKT 2" 63.0 1.050000 "Branch '627205'
'627206' '2'" "OPEN BOTH" "CHECK" "" "" "1bvgr12_successful_reclose"
"Line GARTOW12 TO TOWNSEND 1 CKT 1" 63.0 1.050000 "Branch
'627204' '627201' '1'" "OPEN BOTH" "CHECK" "" ""
"1bvgr12_successful_reclose"
"Line BROADVIEW TO TOWNSEND 1 CKT 1" 63.0 1.050000 "Branch
'623512' '627201' '1'" "OPEN BOTH" "CHECK" "" ""
"1bvgr12_successful_reclose"
"Line BROADVIEW TO TOWNSEND 2 CKT 2" 63.0 1.050000 "Branch
'623512' '627202' '2'" "OPEN BOTH" "CHECK" "" ""
"1bvgr12_successful_reclose"
"Line BROADVIEW TO TOWNSEND 1 CKT 1" 63.0 1.050000 "Branch
'623512' '627201' '1'" "OPEN BOTH" "CHECK" "" ""
"1bvgr12_unsuccessful_reclose"
"Series Cap GAR2EAST TO GARRISON CKT 1" 63.0 1.050000 "Branch
'40453' '40459' '1'" "OPEN BOTH" "CHECK" "" ""
"1bvgr12_unsuccessful_reclose"
"Series Cap GAR1EAST TO GARRISON CKT 1" 111.0 1.850000 "Branch
'40451' '40459' '1'" "OPEN BOTH" "CHECK" "" ""
"1bvgr12_unsuccessful_reclose"
```



```
"Line BROADVIEW TO TOWNSEND 2 CKT 2" 111.0 1.850000 "Branch
'623512' '627202' '2'" "OPEN BOTH" "CHECK" "" ""
"1bvgr12_unsuccessful_reclose"
"Line BROADVIEW TO TOWNSEND 1 CKT 1" 111.0 1.850000 "Branch
'623512' '627201' '1'" "OPEN BOTH" "CHECK" "" ""
"1bvgr12_unsuccessful_reclose"
"Series Cap GAR2EAST TO GARRISON CKT 1" 111.0 1.850000 "Branch
'40453' '40459' '1'" "OPEN BOTH" "CHECK" "" ""
"1bvgr12_unsuccessful_reclose"
"Bus BROADVIEW" 108.0 1.800000 "Bus '623512'" "FAULT SLG ADM 2.579
-35.24" "CHECK" "" "" "1bvgr12_unsuccessful_reclose"
"Line GARTOW11 TO GARTOW12 CKT 1" 63.0 1.050000 "Branch '627203'
'627204' '1'" "OPEN BOTH" "CHECK" "" "" "1bvgr12_unsuccessful_reclose"
"Line GARTOW22 TO TOWNSEND 2 CKT 2" 108.0 1.800000 "Branch
'627206' '627202' '2'" "CLOSE BOTH" "CHECK" "" ""
"1bvgr12_unsuccessful_reclose"
"Line GARTOW21 TO GARTOW22 CKT 2" 108.0 1.800000 "Branch '627205'
'627206' '2'" "CLOSE BOTH" "CHECK" "" ""
"1bvgr12_unsuccessful_reclose"
"Line GARTOW11 TO GARTOW12 CKT 1" 108.0 1.800000 "Branch '627203'
'627204' '1'" "CLOSE BOTH" "CHECK" "" ""
"1bvgr12_unsuccessful_reclose"
"Line BROADVIEW TO TOWNSEND 2 CKT 2" 108.0 1.800000 "Branch
'623512' '627202' '2'" "CLOSE BOTH" "CHECK" "" ""
"1bvgr12_unsuccessful_reclose"
"Line BROADVIEW TO TOWNSEND 1 CKT 1" 108.0 1.800000 "Branch
'623512' '627201' '1'" "CLOSE BOTH" "CHECK" "" ""
"1bvgr12_unsuccessful_reclose"
"Series Cap GAR2EAST TO GARRISON CKT 1" 108.0 1.800000 "Branch
'40453' '40459' '1'" "CLOSE BOTH" "CHECK" "" ""
"1bvgr12_unsuccessful_reclose"
"Line GAR1EAST TO GARTOW11 CKT 1" 108.0 1.800000 "Branch '40451'
'627203' '1'" "CLOSE BOTH" "CHECK" "" ""
"1bvgr12_unsuccessful_reclose"
"Series Cap GAR1EAST TO GARRISON CKT 1" 108.0 1.800000 "Branch
'40451' '40459' '1'" "CLOSE BOTH" "CHECK" "" ""
"1bvgr12_unsuccessful_reclose"
"Line GARTOW22 TO TOWNSEND 2 CKT 2" 63.0 1.050000 "Branch
'627206' '627202' '2'" "OPEN BOTH" "CHECK" "" ""
"1bvgr12_unsuccessful_reclose"
"Line GARTOW21 TO GARTOW22 CKT 2" 63.0 1.050000 "Branch '627205'
'627206' '2'" "OPEN BOTH" "CHECK" "" "" "1bvgr12_unsuccessful_reclose"
"Line GARTOW12 TO TOWNSEND 1 CKT 1" 63.0 1.050000 "Branch
'627204' '627201' '1'" "OPEN BOTH" "CHECK" "" ""
"1bvgr12_unsuccessful_reclose"
"Line BROADVIEW TO TOWNSEND 2 CKT 2" 63.0 1.050000 "Branch
'623512' '627202' '2'" "OPEN BOTH" "CHECK" "" ""
"1bvgr12_unsuccessful_reclose"
"Line GAR2EAST TO GARTOW21 CKT 2" 63.0 1.050000 "Branch '40453'
'627205' '2'" "OPEN BOTH" "CHECK" "" "" "1bvgr12_unsuccessful_reclose"
```



"Bus BROADVIEW" 111.0 1.850000 "Bus '623512'" "CLEARFAULT" "CHECK"
"" "" "1bvgr12_unsuccessful_reclose"
"Line GAR1EAST TO GARTOW11 CKT 1" 63.0 1.050000 "Branch '40451'
'627203' '1'" "OPEN BOTH" "CHECK" "" "" "1bvgr12_unsuccessful_reclose"
"Series Cap GAR1EAST TO GARRISON CKT 1" 63.0 1.050000 "Branch
'40451' '40459' '1'" "OPEN BOTH" "CHECK" "" ""
"1bvgr12_unsuccessful_reclose"
"Bus BROADVIEW" 60.0 1.000000 "Bus '623512'" "FAULT SLG ADM 2.579
-35.24" "CHECK" "" "" "1bvgr12_unsuccessful_reclose"
"Line GARTOW12 TO TOWNSEND 1 CKT 1" 108.0 1.800000 "Branch
'627204' '627201' '1'" "CLOSE BOTH" "CHECK" "" ""
"1bvgr12_unsuccessful_reclose"
"Line GAR2EAST TO GARTOW21 CKT 2" 108.0 1.800000 "Branch '40453'
'627205' '2'" "CLOSE BOTH" "CHECK" "" ""
"1bvgr12_unsuccessful_reclose"
"Bus BROADVIEW" 63.0 1.050000 "Bus '623512'" "CLEARFAULT" "CHECK"
"" "" "1bvgr12_unsuccessful_reclose"
"Series Cap BROADVIEW A TO BROADVIEW CKT 1" 63.0 1.050000
"Branch '623309' '623512' '1'" "OPEN BOTH" "CHECK" "" "" "1csbv1"
"Line COLSTRIP 500 TO BROADVIEW A CKT 1" 63.0 1.050000 "Branch
'623207' '623309' '1'" "OPEN BOTH" "CHECK" "" "" "1csbv1"
"Bus COLSTRIP 500" 60.0 1.000000 "Bus '623207'" "FAULT SLG ADM
2.48 -74.7" "CHECK" "" "" "1csbv1"
"Bus COLSTRIP 500" 63.0 1.050000 "Bus '623207'" "CLEARFAULT"
"CHECK" "" "" "1csbv1"
"Series Cap GARRISON TO GARTAF21 CKT 2" 63.0 1.050000 "Branch
'40459' '41072' '2'" "OPEN BOTH" "CHECK" "" "" "1grtt12"
"Series Cap GARRISON TO GARTAF11 CKT 1" 63.0 1.050000 "Branch
'40459' '41068' '1'" "OPEN BOTH" "CHECK" "" "" "1grtt12"
"Bus GARRISON" 60.0 1.000000 "Bus '40459'" "FAULT SLG ADM 2.66 -
24.4" "CHECK" "" "" "1grtt12"
"Bus GARRISON" 63.0 1.050000 "Bus '40459'" "CLEARFAULT" "CHECK" ""
"" "1grtt12"
"Line GARTAF11 TO GARTAF12 CKT 1" 63.0 1.050000 "Branch '41068'
'41070' '1'" "OPEN BOTH" "CHECK" "" "" "1grtt12"
"Line GARTAF22 TO TAFT CKT 2" 63.0 1.050000 "Branch '41074'
'41057' '2'" "OPEN BOTH" "CHECK" "" "" "1grtt12"
"Line GARTAF21 TO GARTAF22 CKT 2" 63.0 1.050000 "Branch '41072'
'41074' '2'" "OPEN BOTH" "CHECK" "" "" "1grtt12"
"Line GARTAF12 TO TAFT CKT 1" 63.0 1.050000 "Branch '41070'
'41057' '1'" "OPEN BOTH" "CHECK" "" "" "1grtt12"
"Bus BROADVIEW" 15.0 0.250000 "Bus '623512'" "CLEARFAULT" "CHECK"
"" "" "3bvgr1"
"Bus BROADVIEW" 12.0 0.200000 "Bus '623512'" "FAULT 3PB SOLID"
"CHECK" "" "" "3bvgr1"
"Bus GAR1EAST" 15.0 0.250000 "Bus '40451'" "OPEN" "CHECK" "" ""
"3bvgr1"
"Bus TOWNSEND 1" 15.0 0.250000 "Bus '627201'" "OPEN" "CHECK" "" ""
"3bvgr1"



"Bus BROADVIEW" 12.0 0.200000 "Bus '623512'" "FAULT 3PB SOLID"
"CHECK" "" "" "3bvgr12"
"Bus TOWNSEND 2" 15.0 0.250000 "Bus '627202'" "OPEN" "CHECK" "" ""
"3bvgr12"
"Bus TOWNSEND 1" 15.0 0.250000 "Bus '627201'" "OPEN" "CHECK" "" ""
"3bvgr12"
"Bus BROADVIEW" 15.0 0.250000 "Bus '623512'" "CLEARFAULT" "CHECK"
"" "" "3bvgr12"
"Bus GAR2EAST" 15.0 0.250000 "Bus '40453'" "OPEN" "CHECK" "" ""
"3bvgr12"
"Bus GAR1EAST" 15.0 0.250000 "Bus '40451'" "OPEN" "CHECK" "" ""
"3bvgr12"
"Bus BROADVIEW A" 15.0 0.250000 "Bus '623309'" "OPEN" "CHECK" ""
"" "3csbv1"
"Bus COLSTRIP 500" 15.0 0.250000 "Bus '623207'" "CLEARFAULT"
"CHECK" "" "" "3csbv1"
"Bus COLSTRIP 500" 12.0 0.200000 "Bus '623207'" "FAULT 3PB SOLID"
"CHECK" "" "" "3csbv1"
"Bus BROADVIEW B" 15.0 0.250000 "Bus '623310'" "OPEN" "CHECK" ""
"" "3csbv12"
"Bus BROADVIEW A" 15.0 0.250000 "Bus '623309'" "OPEN" "CHECK" ""
"" "3csbv12"
"Bus COLSTRIP 500" 15.0 0.250000 "Bus '623207'" "CLEARFAULT"
"CHECK" "" "" "3csbv12"
"Bus COLSTRIP 500" 12.0 0.200000 "Bus '623207'" "FAULT 3PB SOLID"
"CHECK" "" "" "3csbv12"
"Bus GARRISON" 15.0 0.250000 "Bus '40459'" "CLEARFAULT" "CHECK" ""
"" "3grtt1"
"Series Cap GARRISON TO GARTAF11 CKT 1" 15.0 0.250000 "Branch
'40459' '41068' '1'" "OPEN BOTH" "CHECK" "" "" "3grtt1"
"Bus GARRISON" 12.0 0.200000 "Bus '40459'" "FAULT 3PB SOLID"
"CHECK" "" "" "3grtt1"
"Bus GARRISON" 15.0 0.250000 "Bus '40459'" "CLEARFAULT" "CHECK" ""
"" "3grtt12"
"Series Cap GARRISON TO GARTAF21 CKT 2" 15.0 0.250000 "Branch
'40459' '41072' '2'" "OPEN BOTH" "CHECK" "" "" "3grtt12"
"Series Cap GARRISON TO GARTAF11 CKT 1" 15.0 0.250000 "Branch
'40459' '41068' '1'" "OPEN BOTH" "CHECK" "" "" "3grtt12"
"Bus GARRISON" 12.0 0.200000 "Bus '40459'" "FAULT 3PB SOLID"
"CHECK" "" "" "3grtt12"
"Gen LIB 05 #5" 23.0 0.383330 "Gen '44195' '5'" "OPEN" "CHECK" ""
"" "3ttbb"
"Gen LIB 04 #4" 23.0 0.383330 "Gen '44194' '4'" "OPEN" "CHECK" ""
"" "3ttbb"
"Gen LIB 03 #3" 23.0 0.383330 "Gen '44193' '3'" "OPEN" "CHECK" ""
"" "3ttbb"
"Gen LIB 02 #2" 23.0 0.383330 "Gen '44192' '2'" "OPEN" "CHECK" ""
"" "3ttbb"
"Gen LIB 01 #1" 23.0 0.383330 "Gen '44191' '1'" "OPEN" "CHECK" ""
"" "3ttbb"



```
"Shunt GARRISON #r4"      18.5 0.308330 "Shunt '40459' 'r4'" "OPEN"
"CHECK" "" "" "3ttbb"
"Bus TAFT"      15.0 0.250000 "Bus '41057'" "CLEARFAULT" "CHECK" "" ""
"3ttbb"
"Bus BELL SC"    15.0 0.250000 "Bus '40096'" "OPEN" "CHECK" "" ""
"3ttbb"
"Bus TAFT"      12.0 0.200000 "Bus '41057'" "FAULT 3PB SOLID" "CHECK"
"" "" "3ttbb"
"Shunt GARRISON #r3"    18.0 0.300000 "Shunt '40459' 'r3'" "OPEN"
"CHECK" "" "" "bvgr12"
"Line GAR2EAST TO GARTOW21 CKT 2"      12.0 0.200000 "Branch '40453'
'627205' '2'" "OPEN BOTH" "CHECK" "" "" "bvgr12"
"Series Cap GAR2EAST TO GARRISON CKT 1"      12.0 0.200000 "Branch
'40453' '40459' '1'" "OPEN BOTH" "CHECK" "" "" "bvgr12"
"Line GAR1EAST TO GARTOW11 CKT 1"      12.0 0.200000 "Branch '40451'
'627203' '1'" "OPEN BOTH" "CHECK" "" "" "bvgr12"
"Series Cap GAR1EAST TO GARRISON CKT 1"      12.0 0.200000 "Branch
'40451' '40459' '1'" "OPEN BOTH" "CHECK" "" "" "bvgr12"
"Series Cap GARRISON TO GARTAF21 CKT 2"      12.0 0.200000 "Branch
'40459' '41072' '2'" "OPEN BOTH" "CHECK" "" "" "grtt12"
"Series Cap GARRISON TO GARTAF11 CKT 1"      12.0 0.200000 "Branch
'40459' '41068' '1'" "OPEN BOTH" "CHECK" "" "" "grtt12"
}
```

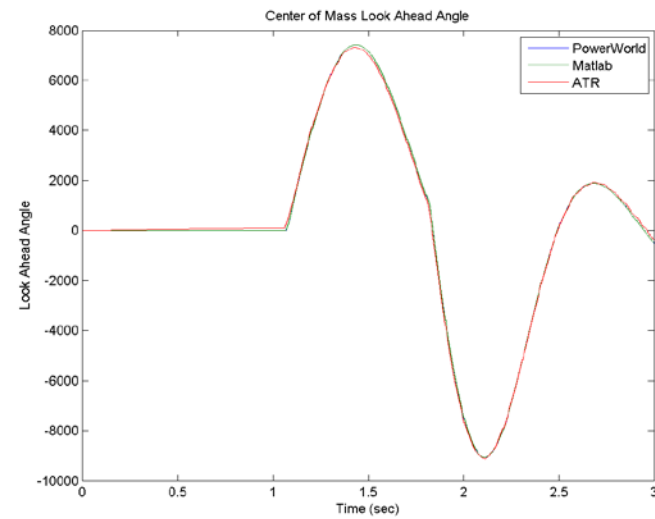
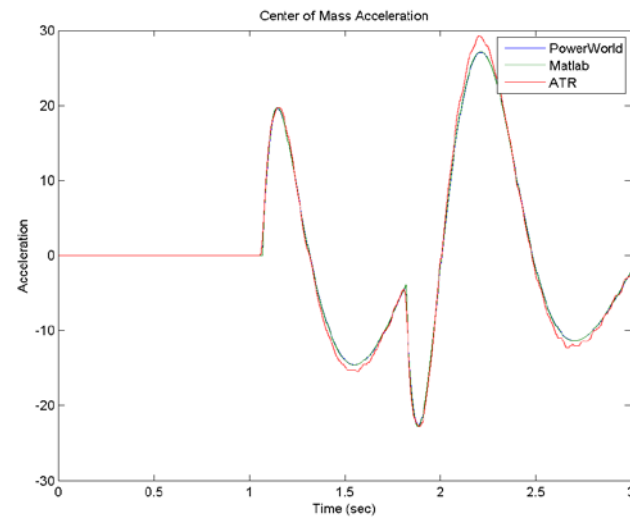
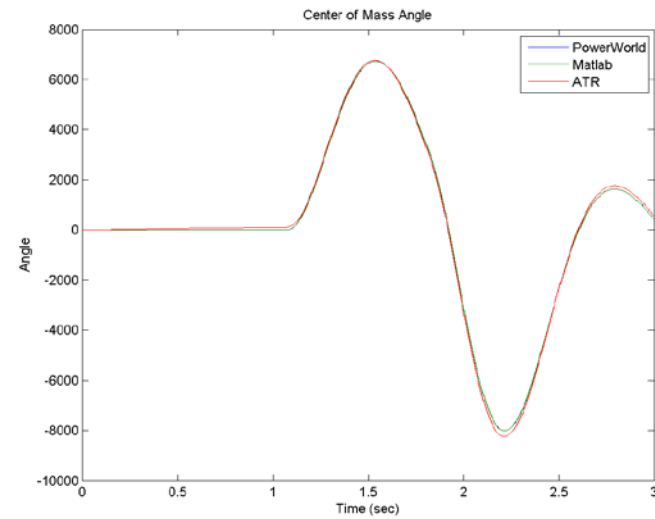
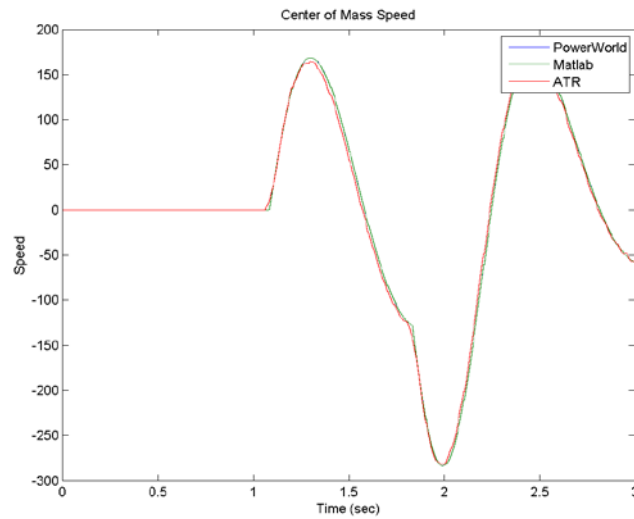
For the purposes of single line openings, the following table gives the 2-phase equivalent of the listed line segments:

Line	From Bus	To Bus	R	X	From Bus Shunt (G)	From Bus Shunt (B)	To Bus Shunt (G)	From Bus Shunt (B)
Broadview – Garrison 1 or 2	623512	40459	0.00559	0.08353	-0.00058	0.37662	0.00093	0.39544

The following tables show the trip times of each unit (if any) and total MW and percentage tripped by each ATR model. The tables show a 3 second window of time of the Center of Mass values for each ATR model.

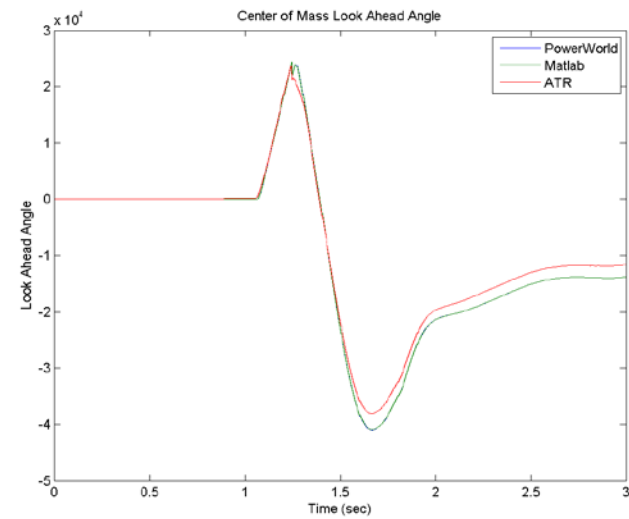
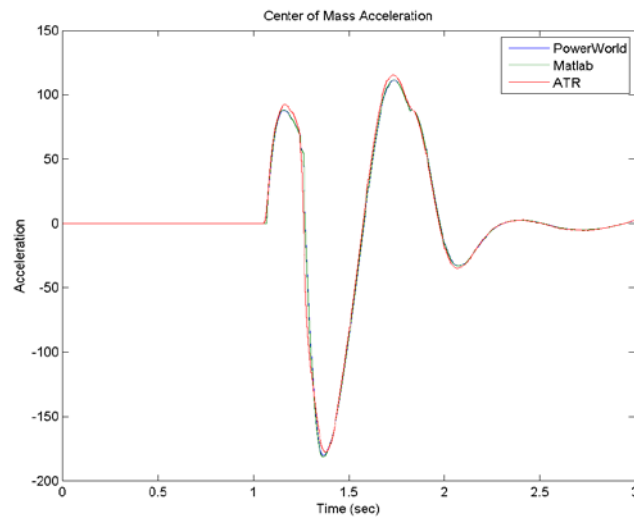
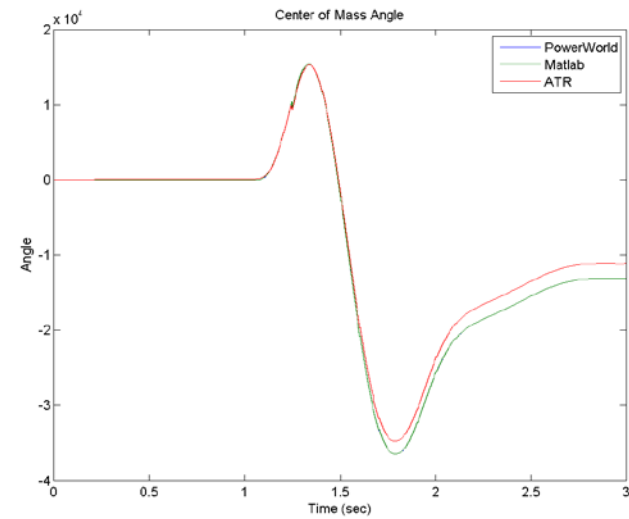
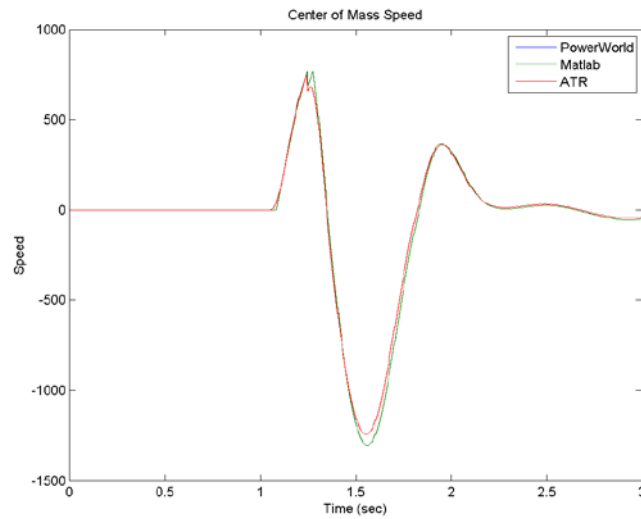
15hs4a_1bvgr1

	Unit 1 Trip Time (sec)	Unit 2 Trip Time (sec)	Unit 3 Trip Time (sec)	Unit 4 Trip Time (sec)	Final MW Tripped	Final % Tripped
ATR	0	0	0	0	0	0
Matlab	0	0	0	0	0	0
PowerWorld	0	0	0	0	0	0



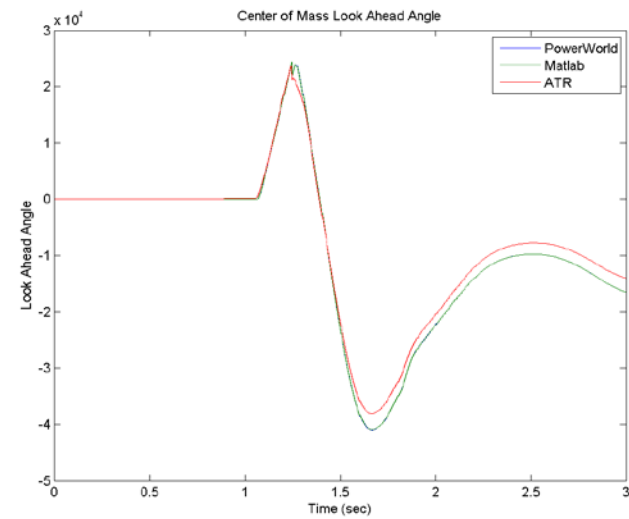
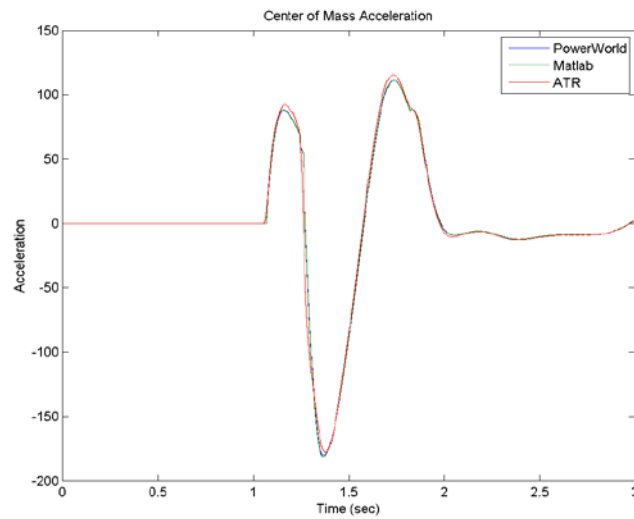
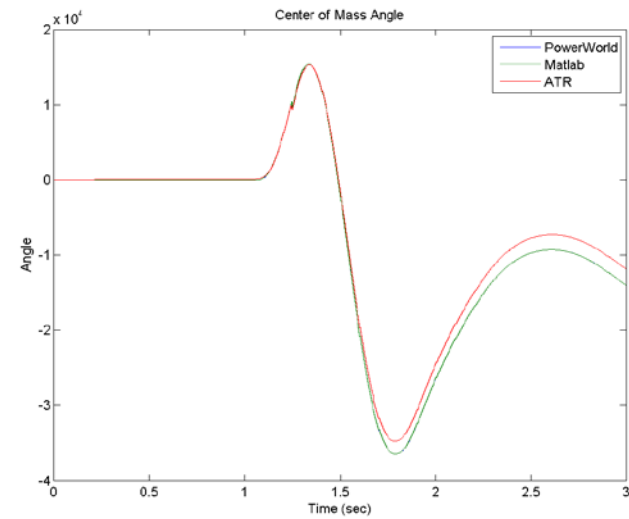
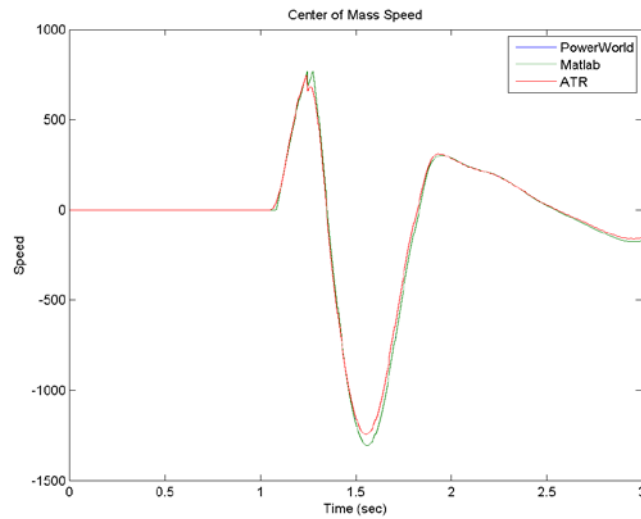
15hs4a_1bvgr12

	Unit 1 Trip Time (sec)	Unit 2 Trip Time (sec)	Unit 3 Trip Time (sec)	Unit 4 Trip Time (sec)	Final MW Tripped	Final % Tripped
ATR	0	0	1.1765	1.1765	1605	71
Matlab	1.1765	1.1932	1.1765	0	1460	64
PowerWorld	1.1765	1.1932	1.1765	0	1460	64



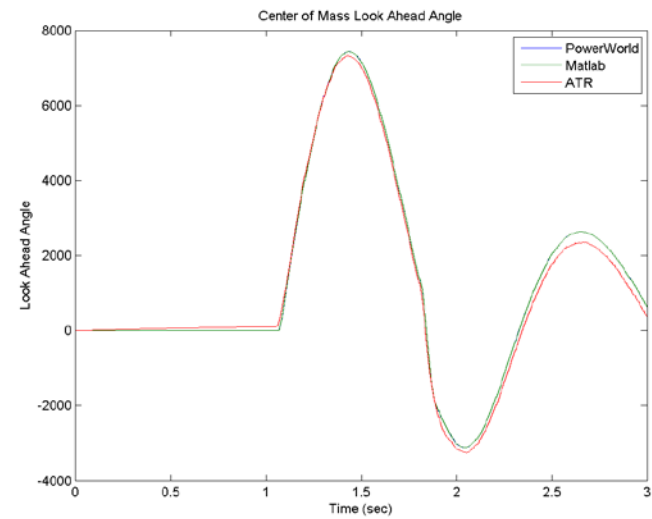
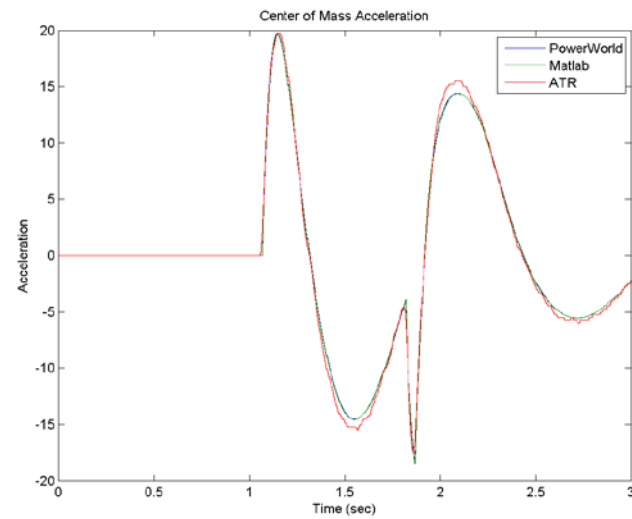
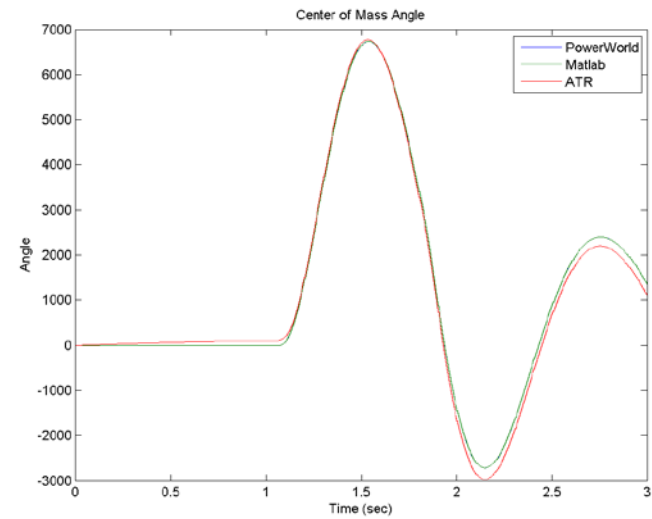
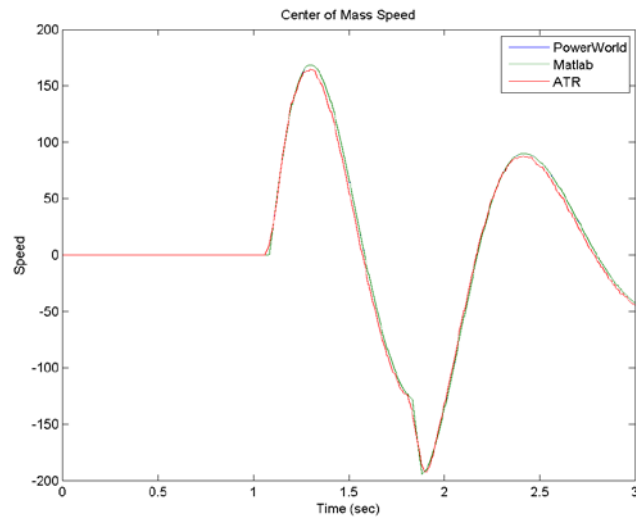
15hs4a_1bvgr12_ur

	Unit 1 Trip Time (sec)	Unit 2 Trip Time (sec)	Unit 3 Trip Time (sec)	Unit 4 Trip Time (sec)	Final MW Tripped	Final % Tripped
ATR	0	0	1.1765	1.1765	1605	71
Matlab	1.1765	1.1932	1.1765	0	1460	64
PowerWorld	1.1765	1.1932	1.1765	0	1460	64



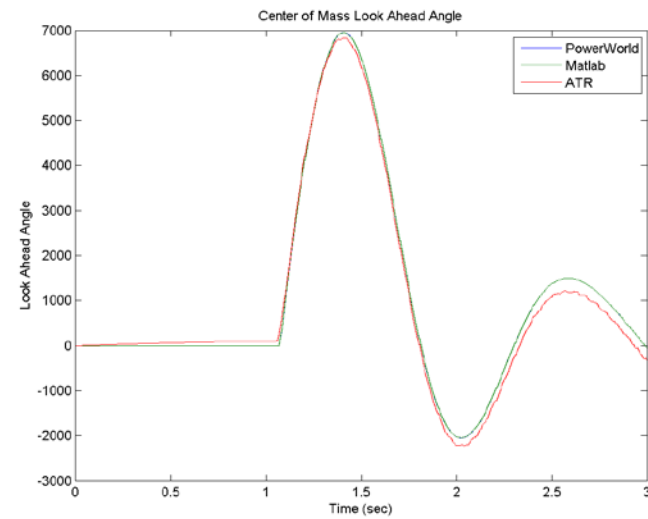
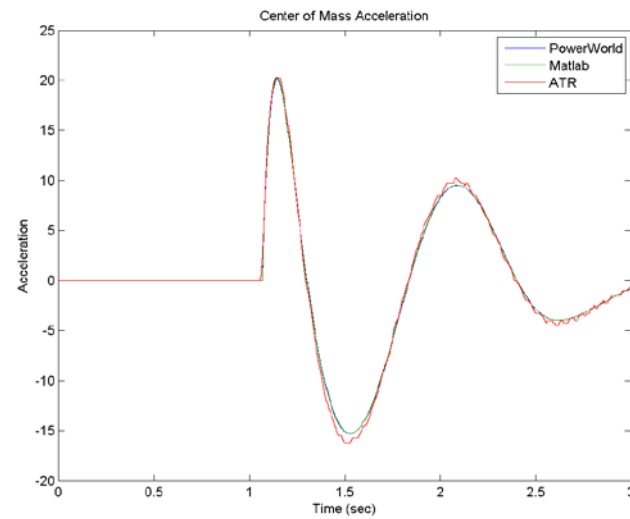
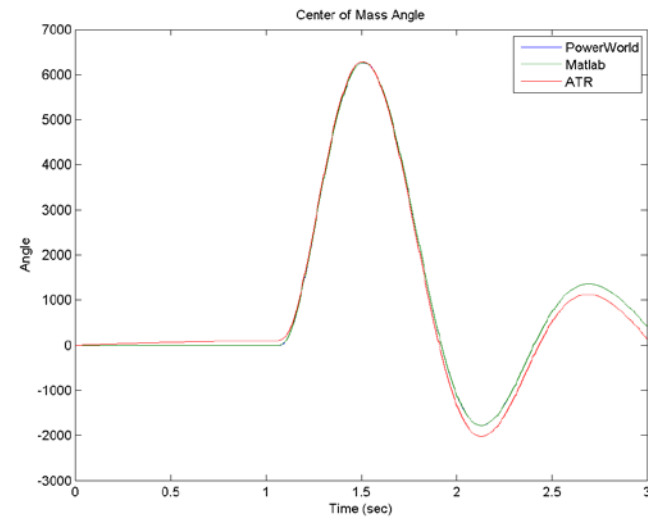
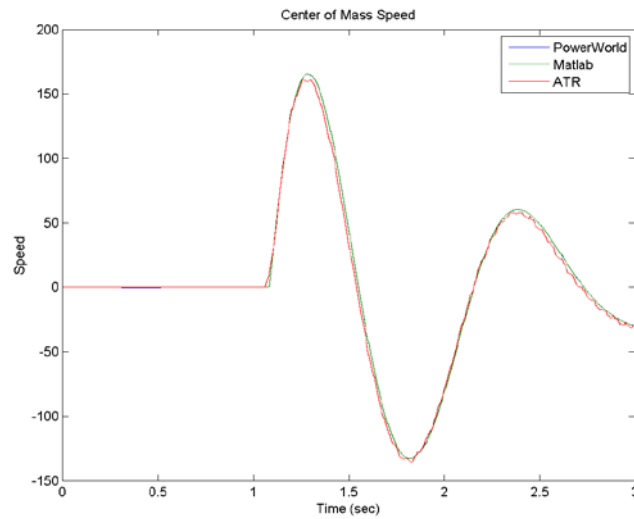
15hs4a_1bvgr1_ur

	Unit 1 Trip Time (sec)	Unit 2 Trip Time (sec)	Unit 3 Trip Time (sec)	Unit 4 Trip Time (sec)	Final MW Tripped	Final % Tripped
ATR	0	0	0	0	0	0
Matlab	0	0	0	0	0	0
PowerWorld	0	0	0	0	0	0



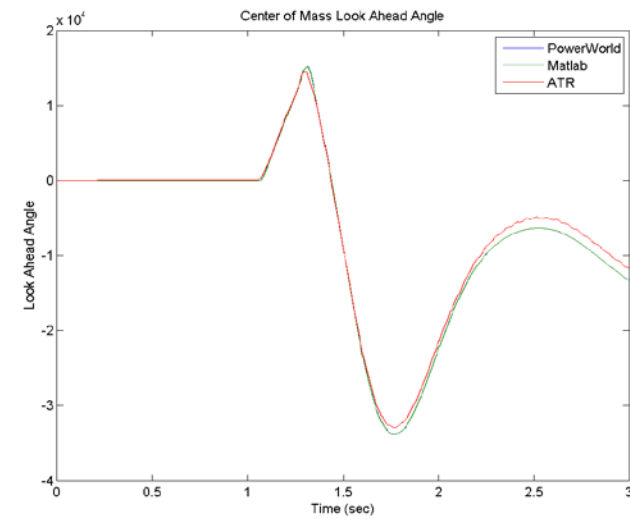
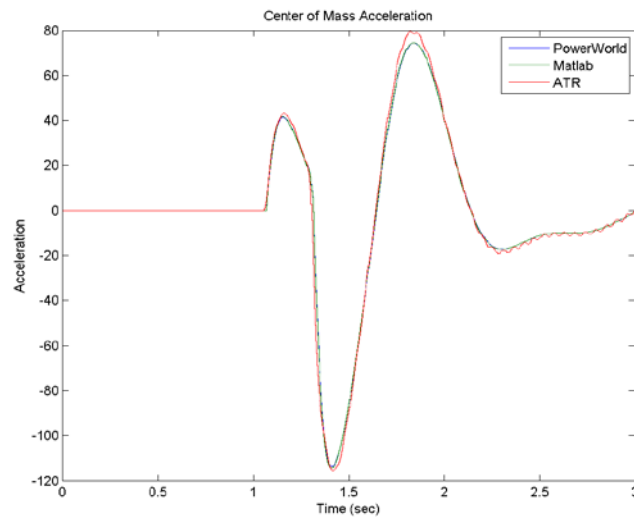
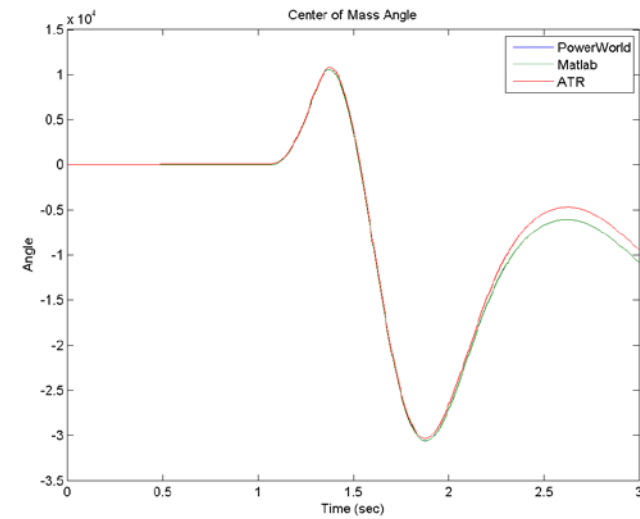
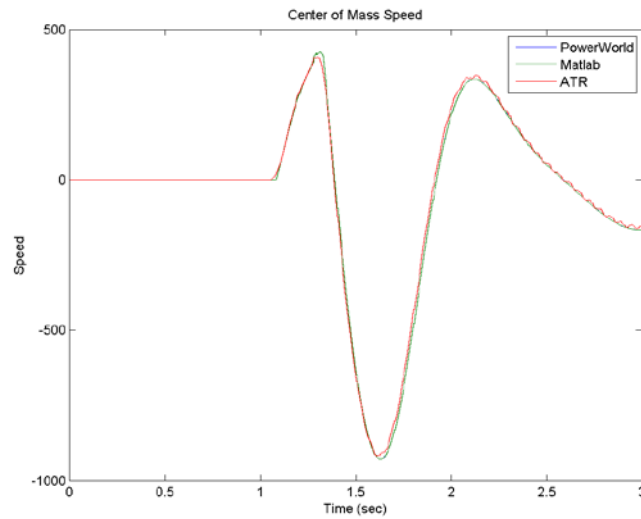
15hs4a_1csbv1

	Unit 1 Trip Time (sec)	Unit 2 Trip Time (sec)	Unit 3 Trip Time (sec)	Unit 4 Trip Time (sec)	Final MW Tripped	Final % Tripped
ATR	0	0	0	0	0	0
Matlab	0	0	0	0	0	0
PowerWorld	0	0	0	0	0	0



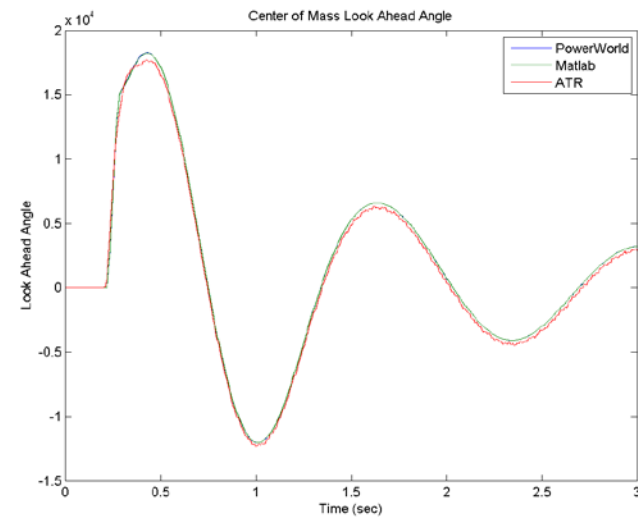
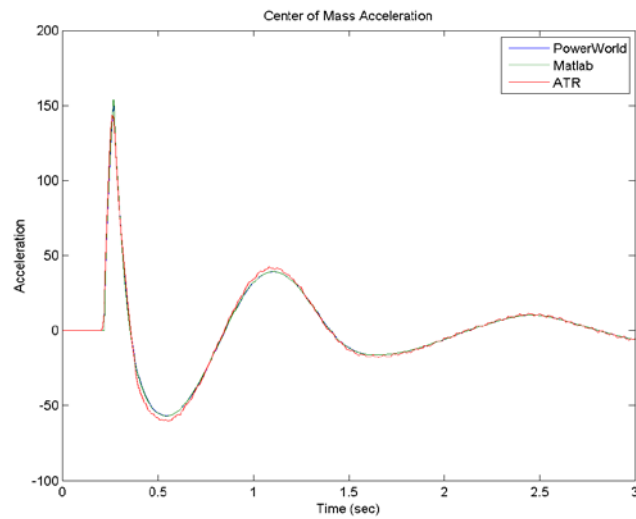
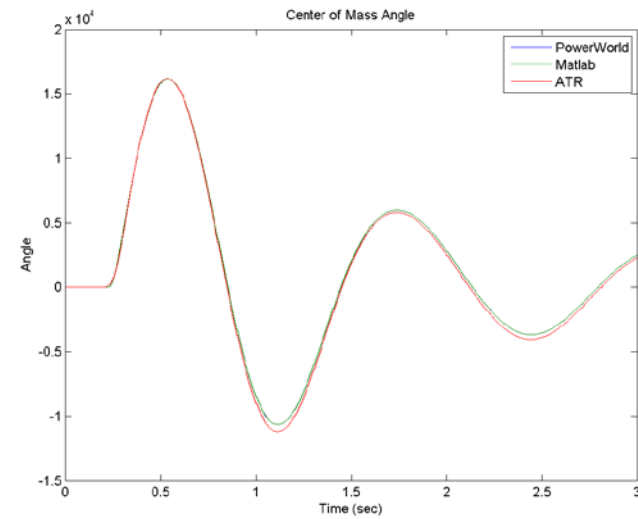
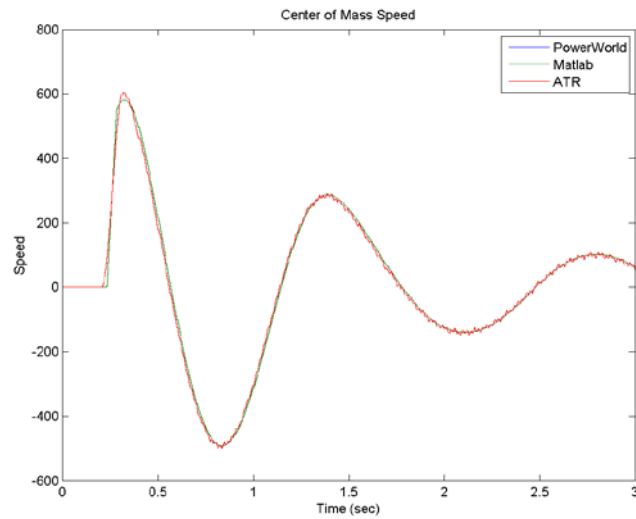
15hs4a_1grtt12

	Unit 1 Trip Time (sec)	Unit 2 Trip Time (sec)	Unit 3 Trip Time (sec)	Unit 4 Trip Time (sec)	Final MW Tripped	Final % Tripped
ATR	1.2299	0	1.2265	0	1130	50
Matlab	1.2265	0	1.2265	0	1130	50
PowerWorld	1.2265	0	1.2265	0	1130	50



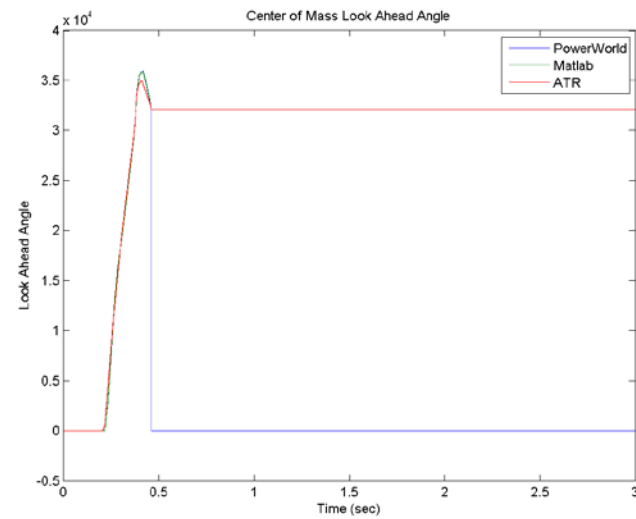
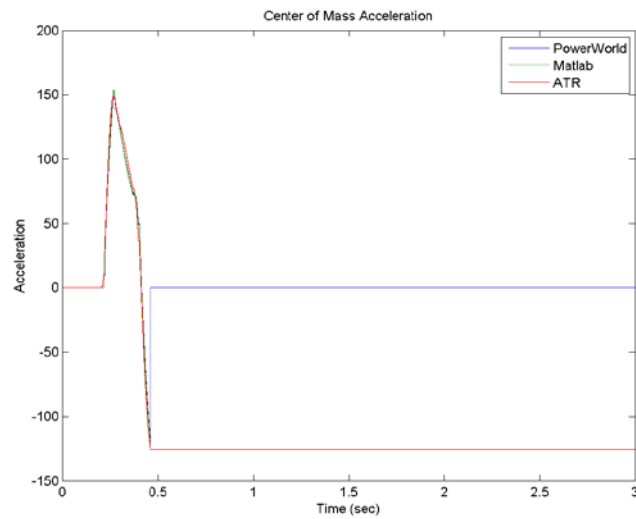
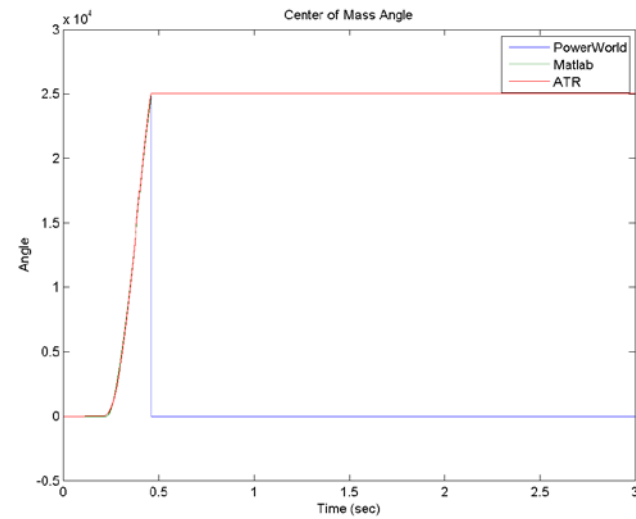
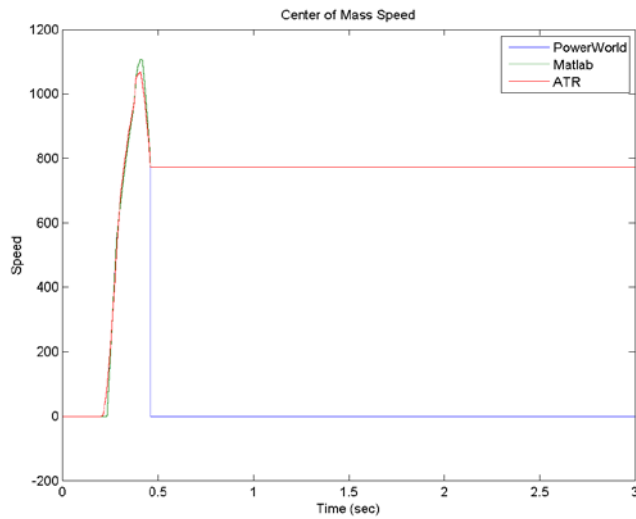
15hs4a_3bvgr1

	Unit 1 Trip Time (sec)	Unit 2 Trip Time (sec)	Unit 3 Trip Time (sec)	Unit 4 Trip Time (sec)	Final MW Tripped	Final % Tripped
ATR	0	0	0	0	0	0
Matlab	0	0	0	0	0	0
PowerWorld	0	0	0	0	0	0



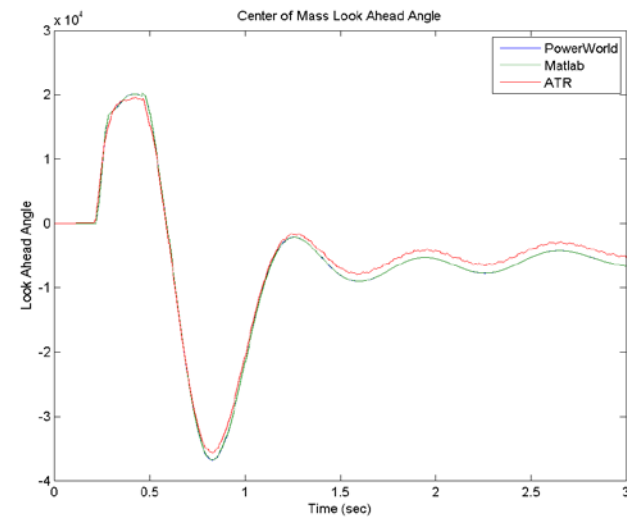
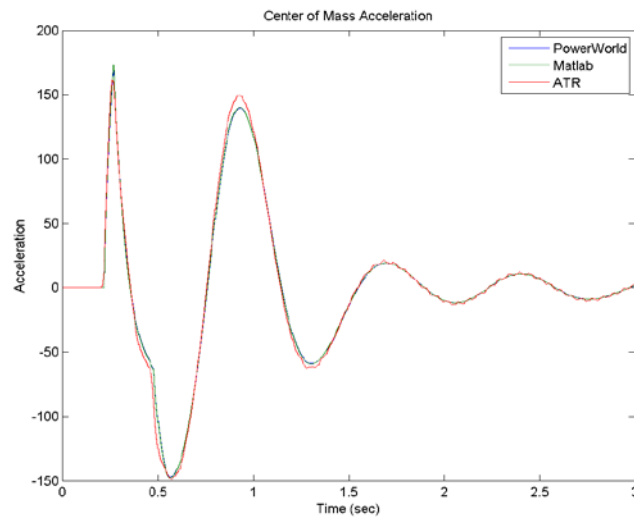
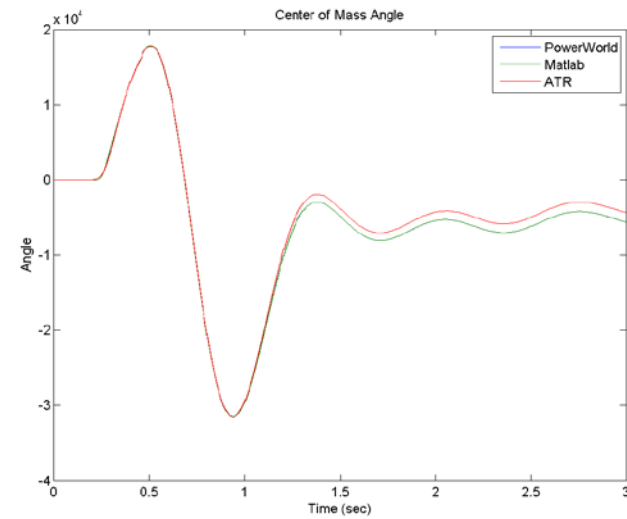
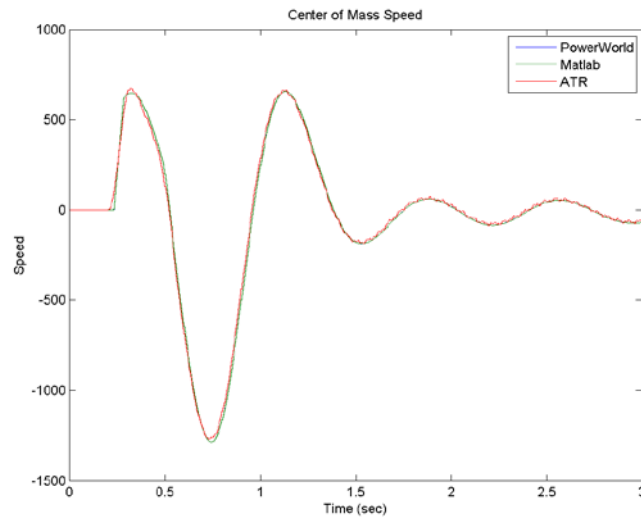
15hs4a_3bvgr12

	Unit 1 Trip Time (sec)	Unit 2 Trip Time (sec)	Unit 3 Trip Time (sec)	Unit 4 Trip Time (sec)	Final MW Tripped	Final % Tripped
ATR	0.30997	0.3233	0.30997	0.39329	2265	100
Matlab	0.31664	0.33997	0.31664	0.39329	2265	100
PowerWorld	0.31664	0.33997	0.31664	0.39329	2265	100



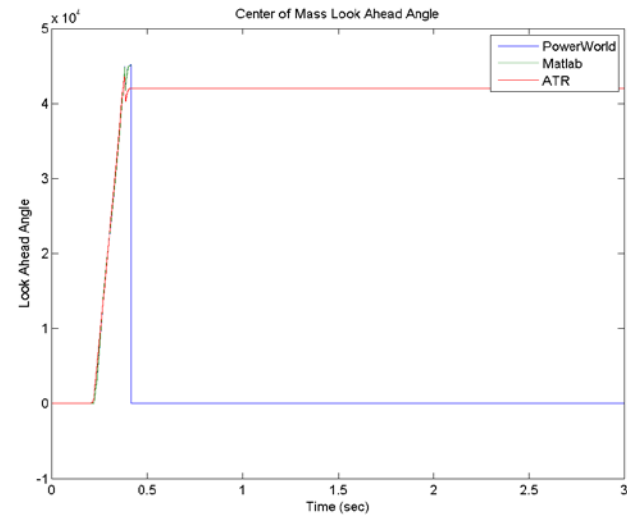
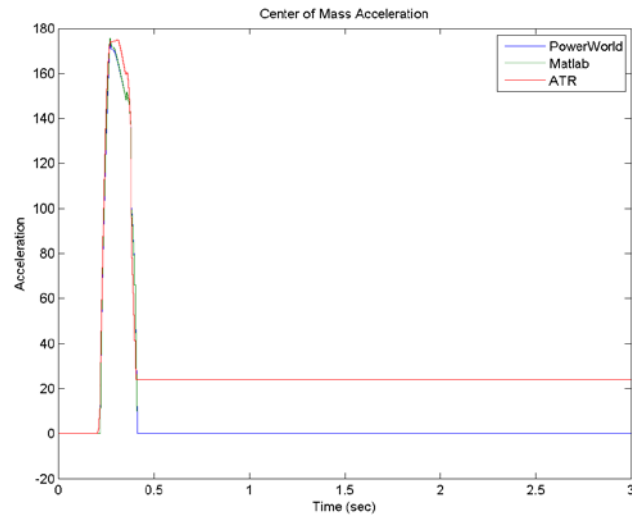
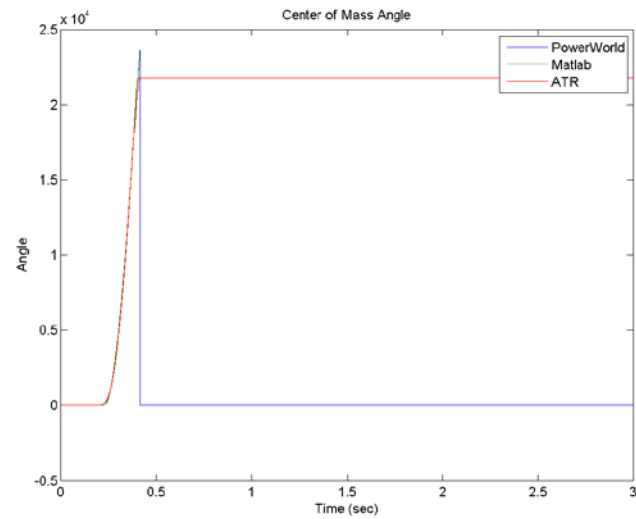
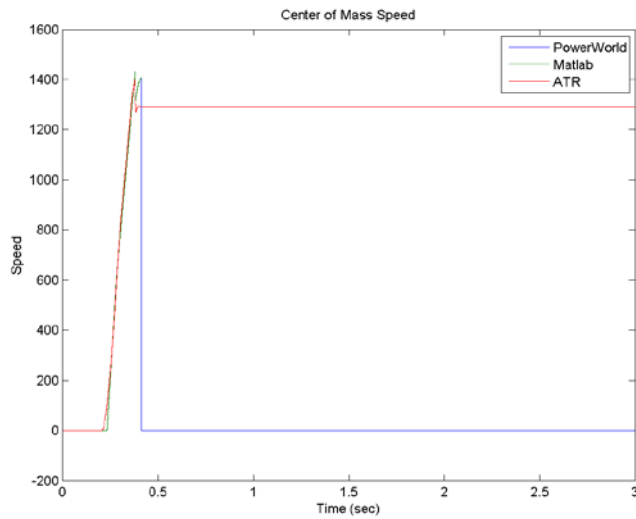
15hs4a_3csbv1

	Unit 1 Trip Time (sec)	Unit 2 Trip Time (sec)	Unit 3 Trip Time (sec)	Unit 4 Trip Time (sec)	Final MW Tripped	Final % Tripped
ATR	0	0	0	0	0	0
Matlab	0	0	0.39329	0	800	35
PowerWorld	0	0	0.39329	0	800	35



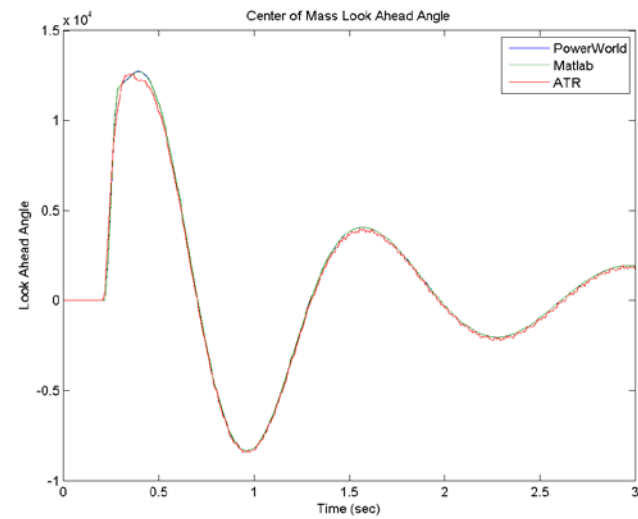
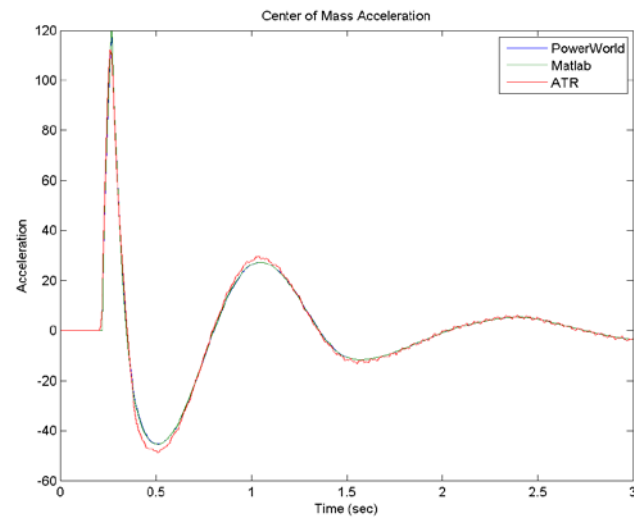
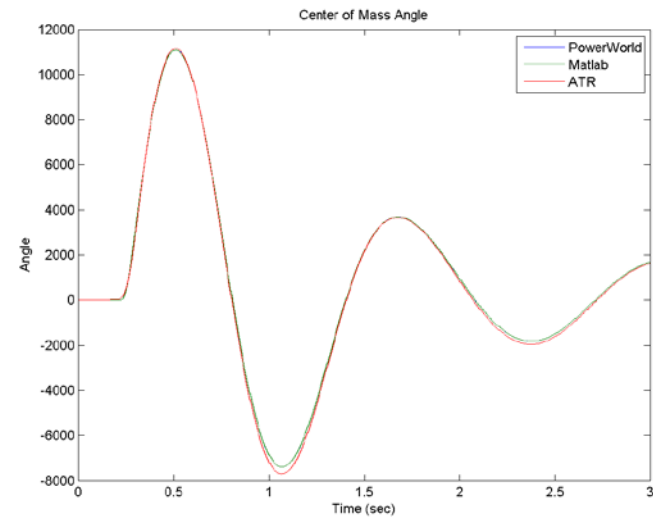
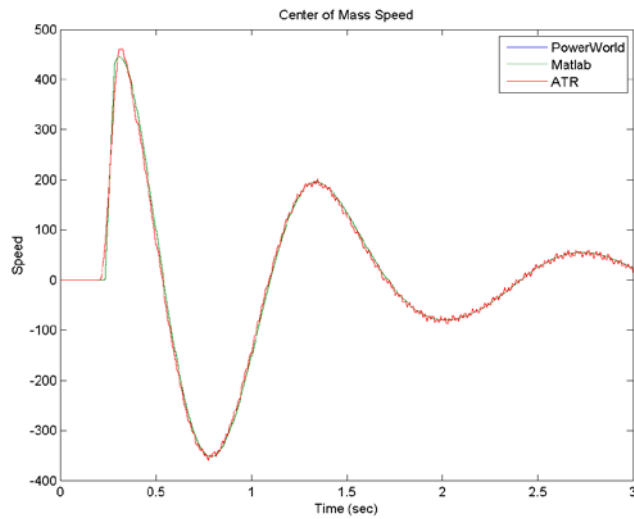
15hs4a_3csbv12

	Unit 1 Trip Time (sec)	Unit 2 Trip Time (sec)	Unit 3 Trip Time (sec)	Unit 4 Trip Time (sec)	Final MW Tripped	Final % Tripped
ATR	0.3033	0.36996	0.3033	0.30664	2265	100
Matlab	0.30664	0.36996	0.30664	0.3133	2265	100
PowerWorld	0.3033	0.36996	0.3033	0.3133	2265	100



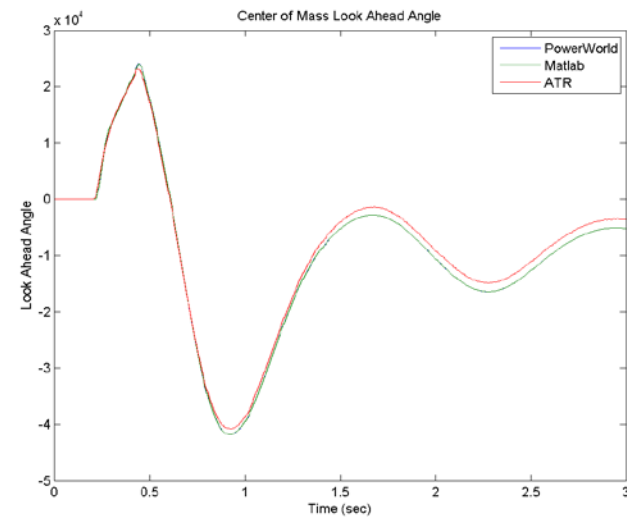
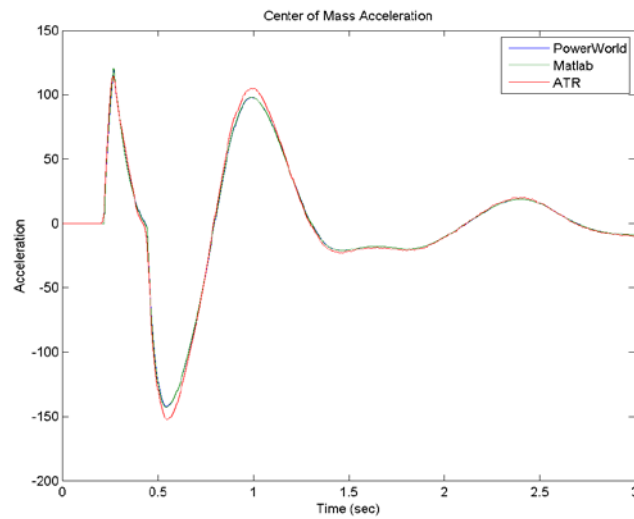
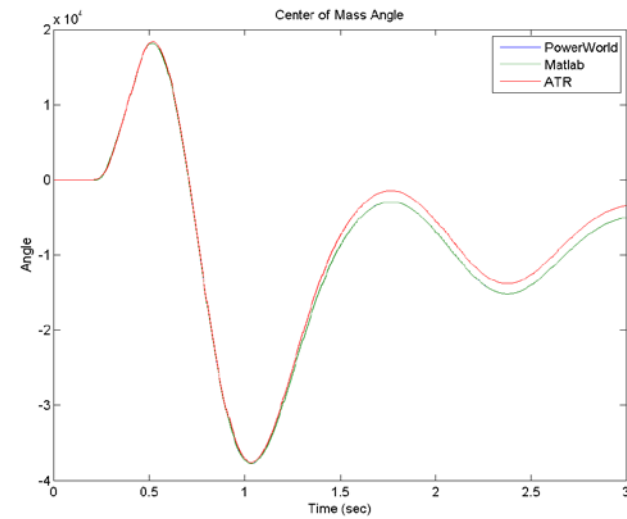
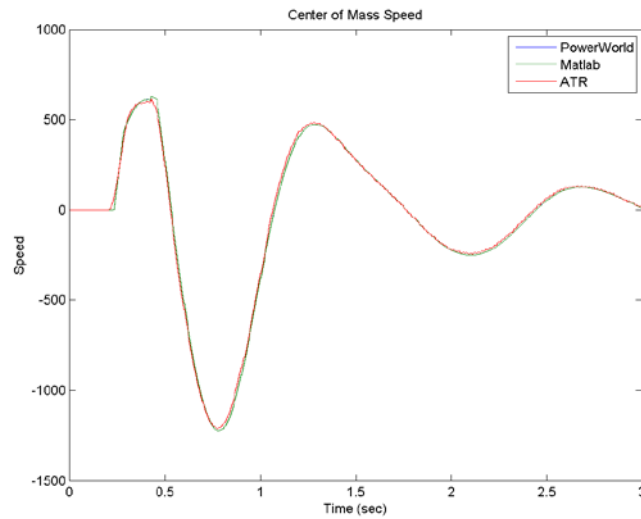
15hs4a_3grtt1

	Unit 1 Trip Time (sec)	Unit 2 Trip Time (sec)	Unit 3 Trip Time (sec)	Unit 4 Trip Time (sec)	Final MW Tripped	Final % Tripped
ATR	0	0	0	0	0	0
Matlab	0	0	0	0	0	0
PowerWorld	0	0	0	0	0	0



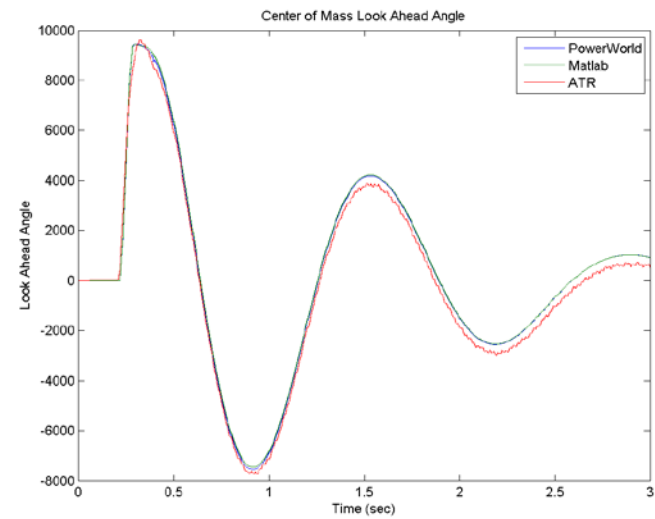
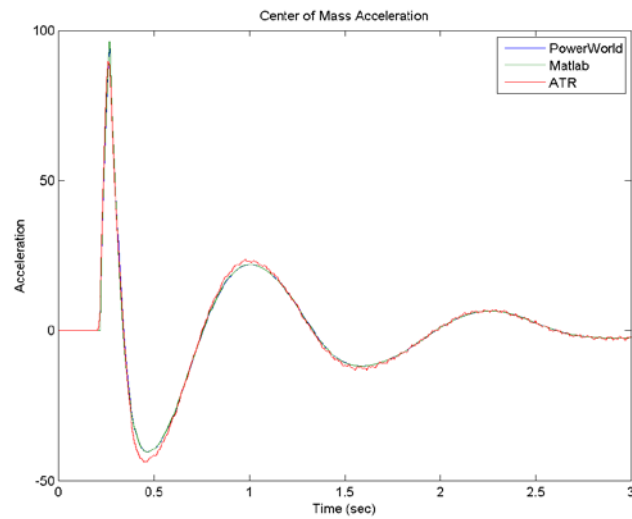
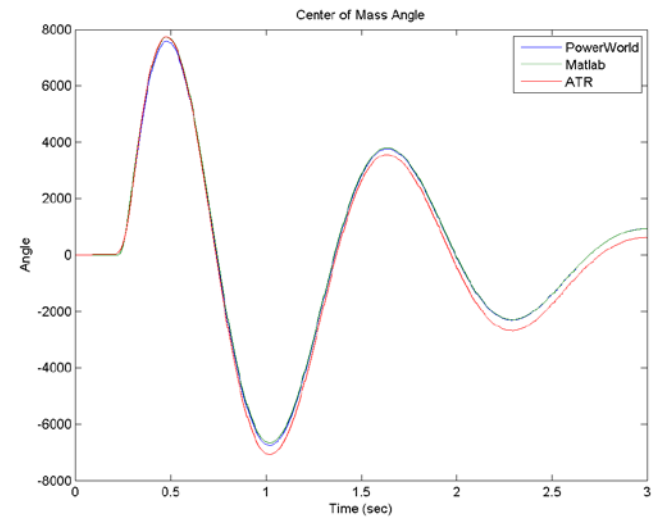
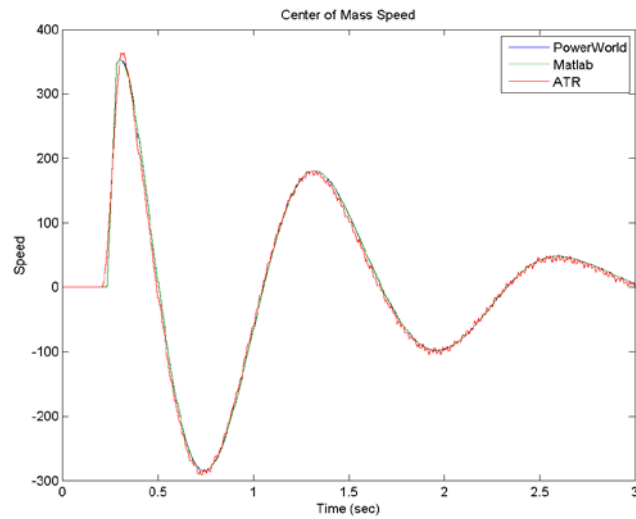
15hs4a_3grtt12

	Unit 1 Trip Time (sec)	Unit 2 Trip Time (sec)	Unit 3 Trip Time (sec)	Unit 4 Trip Time (sec)	Final MW Tripped	Final % Tripped
ATR	0.34663	0	0.34663	0	1130	50
Matlab	0.37996	0	0.35996	0	1130	50
PowerWorld	0.3733	0	0.35996	0	1130	50



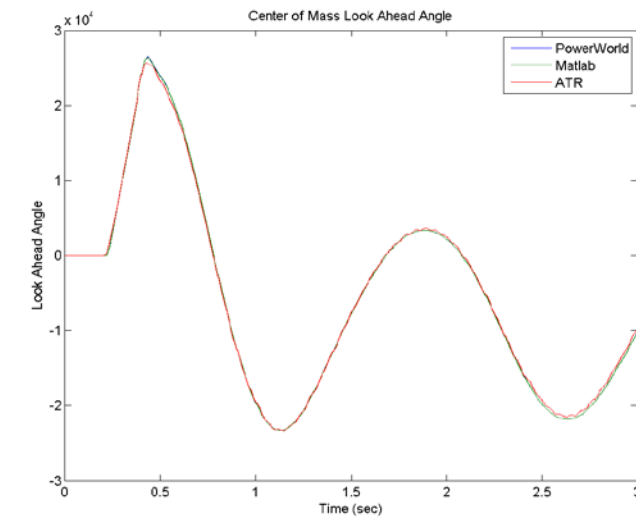
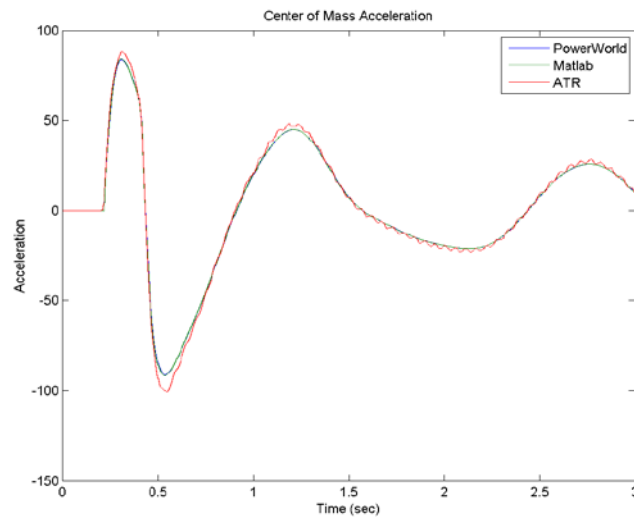
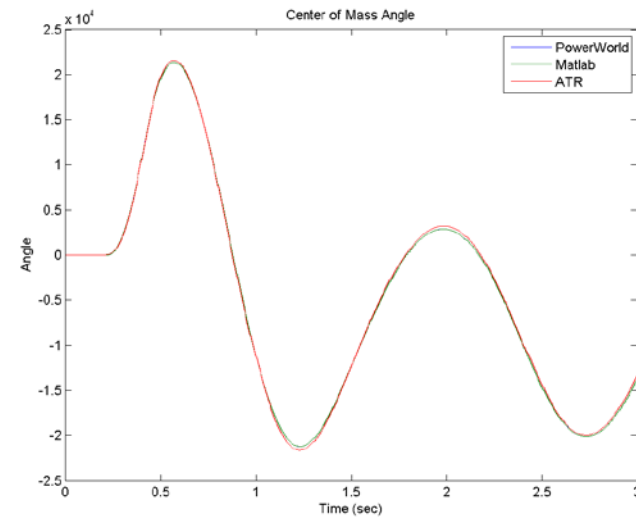
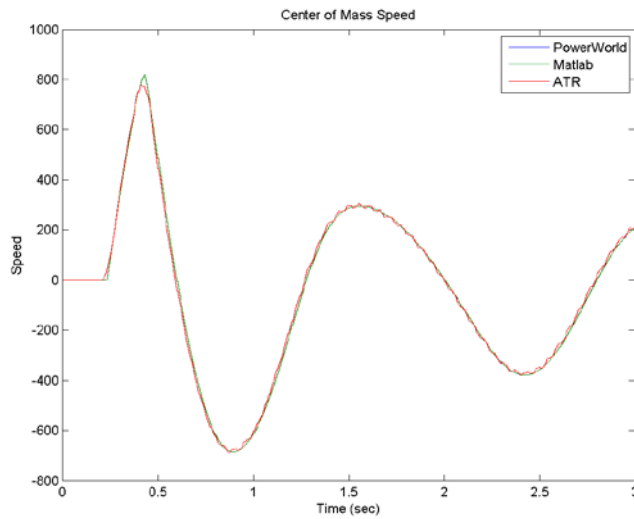
15hs4a_3ttbb

	Unit 1 Trip Time (sec)	Unit 2 Trip Time (sec)	Unit 3 Trip Time (sec)	Unit 4 Trip Time (sec)	Final MW Tripped	Final % Tripped
ATR	0	0	0	0	0	0
Matlab	0	0	0	0	0	0
PowerWorld	0	0	0	0	0	0



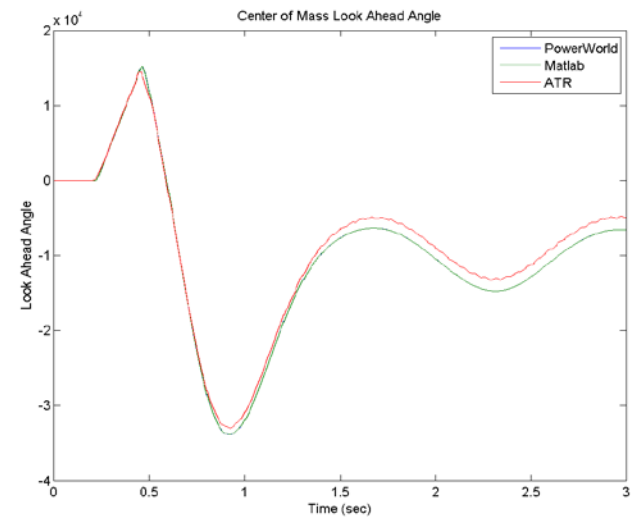
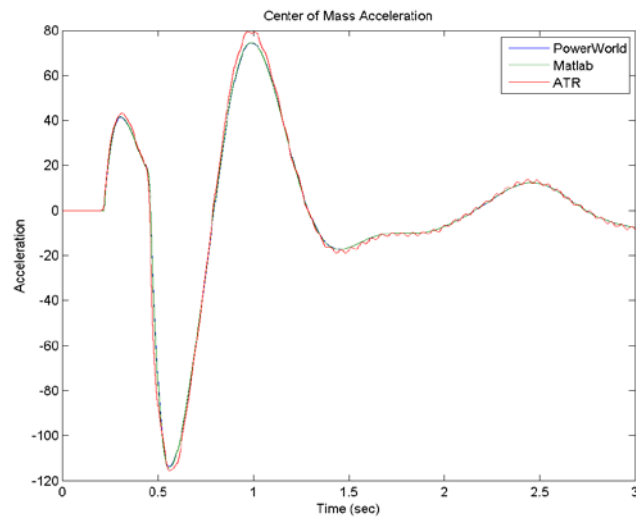
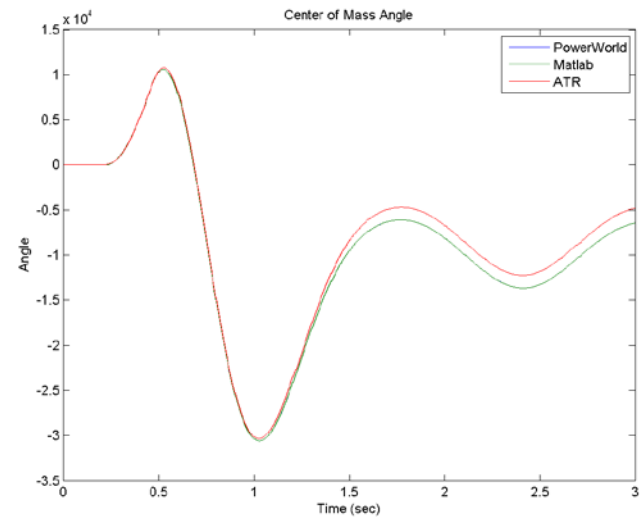
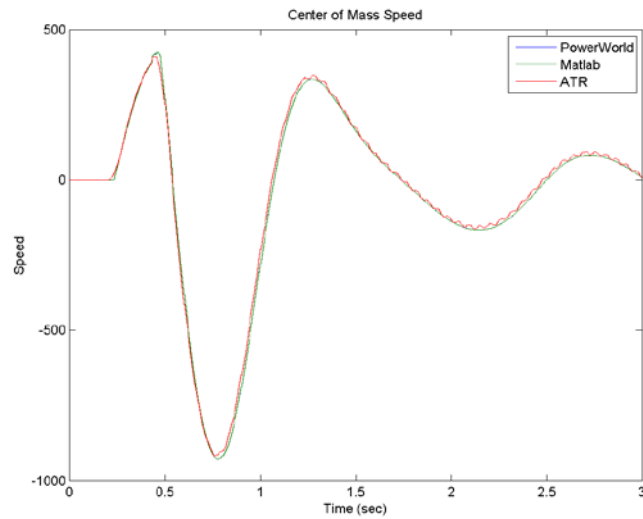
15hs4a_bvgr12

	Unit 1 Trip Time (sec)	Unit 2 Trip Time (sec)	Unit 3 Trip Time (sec)	Unit 4 Trip Time (sec)	Final MW Tripped	Final % Tripped
ATR	0.32997	0.3433	0.32997	0	1460	64
Matlab	0.32997	0	0.32997	0	1130	50
PowerWorld	0.32997	0	0.32997	0	1130	50



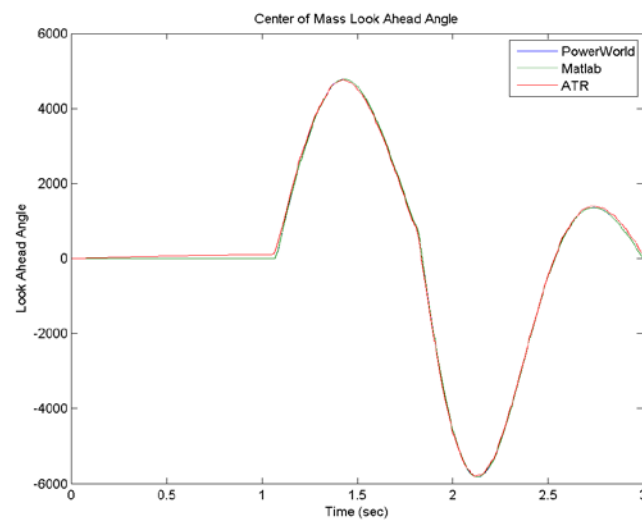
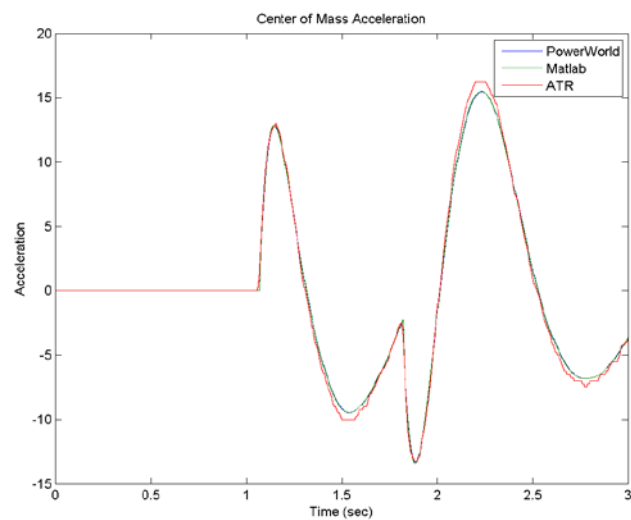
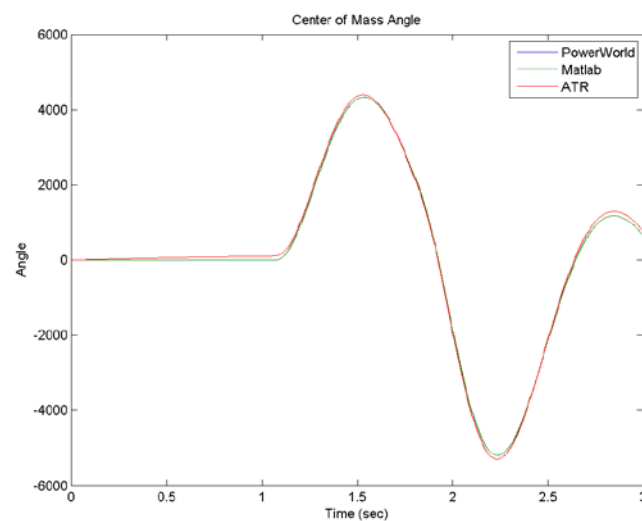
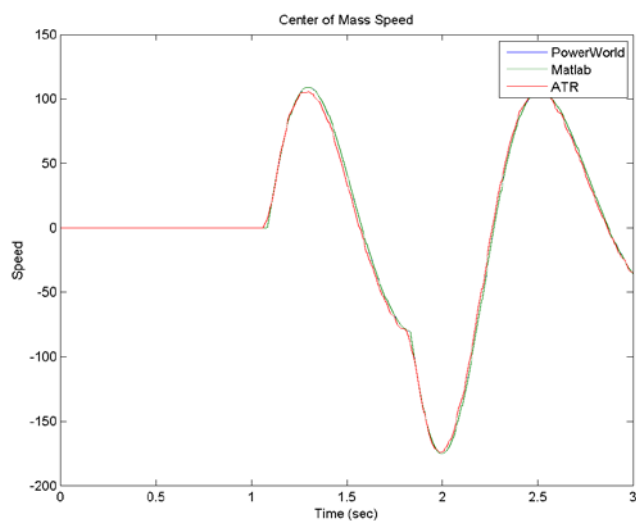
15hs4a_grtt12

	Unit 1 Trip Time (sec)	Unit 2 Trip Time (sec)	Unit 3 Trip Time (sec)	Unit 4 Trip Time (sec)	Final MW Tripped	Final % Tripped
ATR	0.37996	0	0.37663	0	1130	50
Matlab	0.37663	0	0.37663	0	1130	50
PowerWorld	0.37663	0	0.37663	0	1130	50



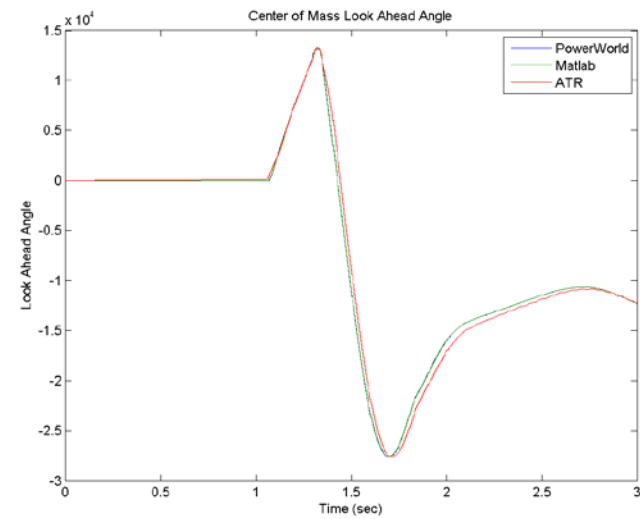
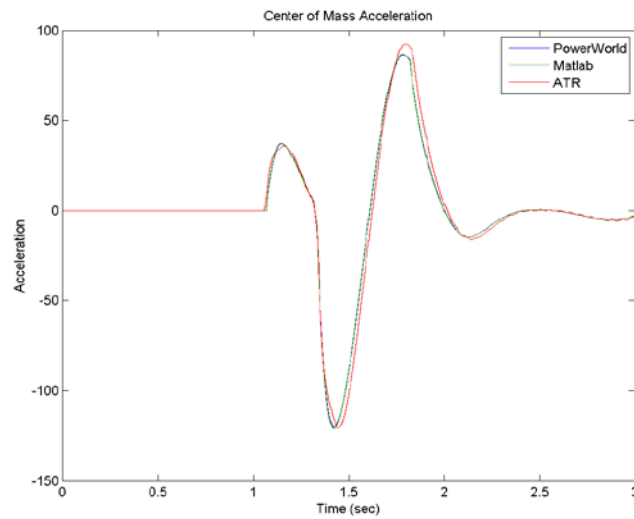
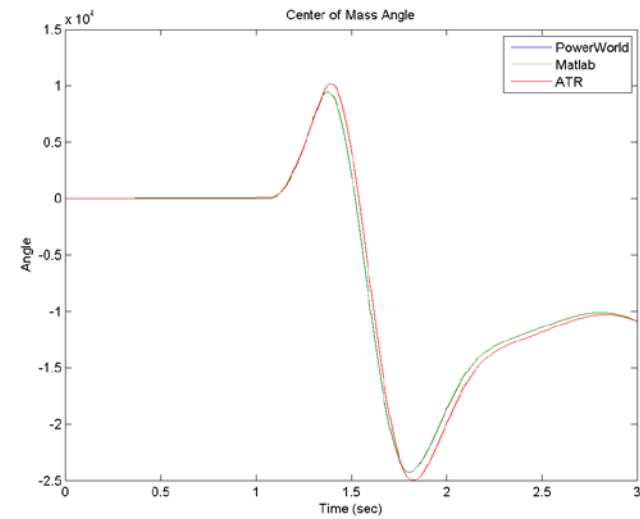
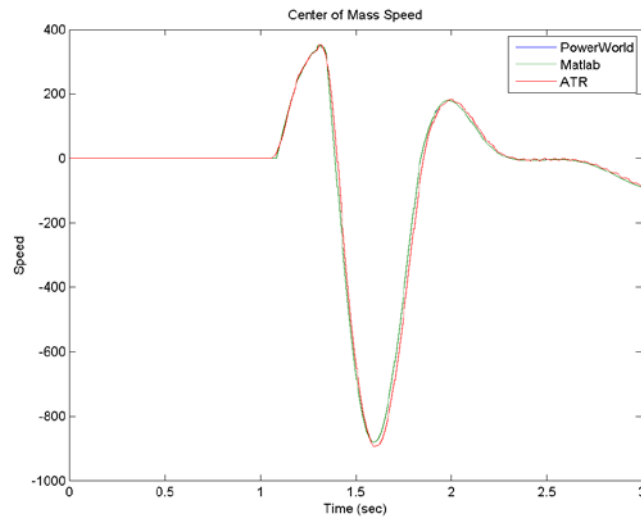
15ls1a_1bvgr1

	Unit 1 Trip Time (sec)	Unit 2 Trip Time (sec)	Unit 3 Trip Time (sec)	Unit 4 Trip Time (sec)	Final MW Tripped	Final % Tripped
ATR	0	0	0	0	0	0
Matlab	0	0	0	0	0	0
PowerWorld	0	0	0	0	0	0



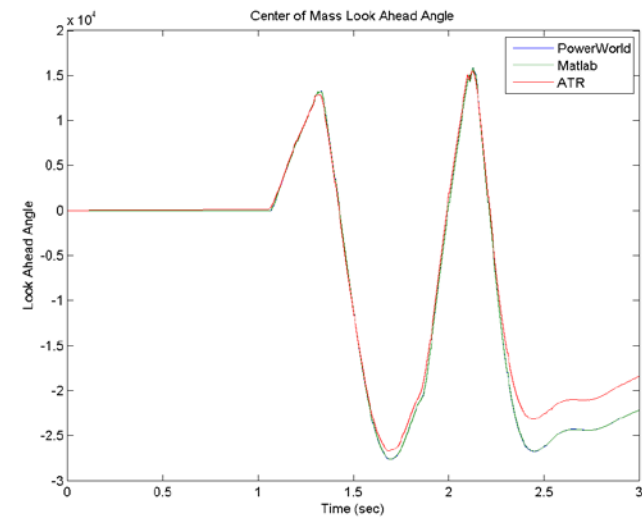
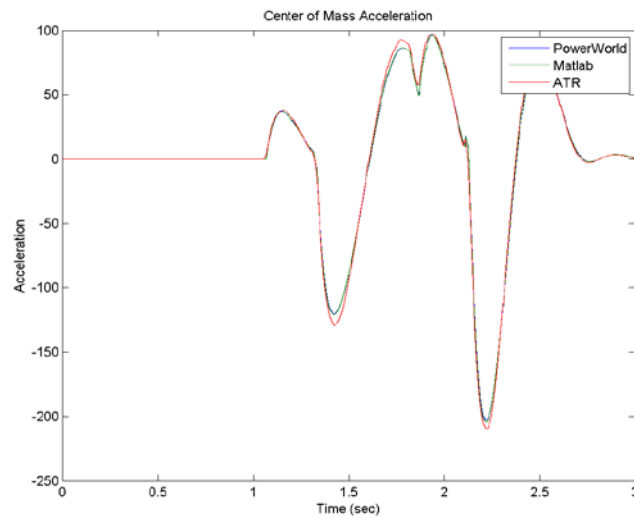
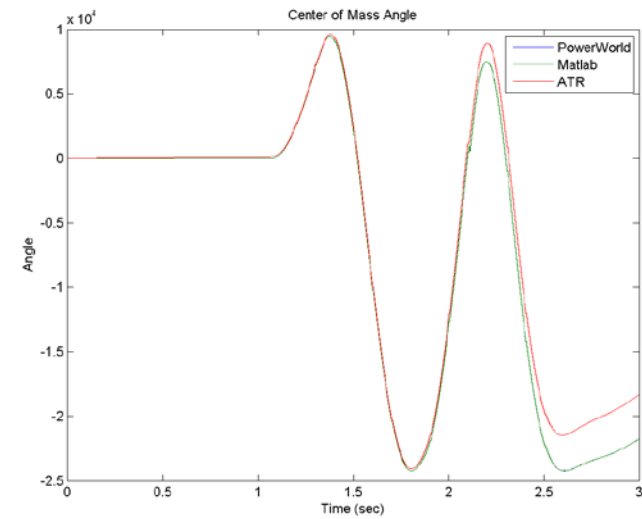
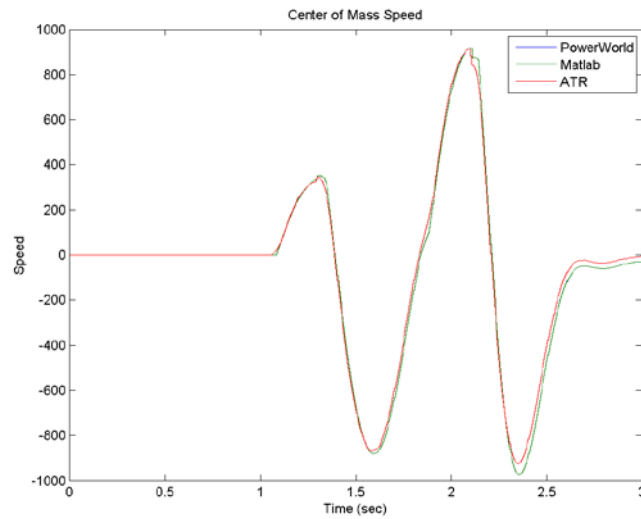
15ls1a_1bvgr12

	Unit 1 Trip Time (sec)	Unit 2 Trip Time (sec)	Unit 3 Trip Time (sec)	Unit 4 Trip Time (sec)	Final MW Tripped	Final % Tripped
ATR	1.2399	0	1.2365	0	1125	50
Matlab	1.2465	0	1.2465	0	1125	50
PowerWorld	1.2465	0	1.2465	0	1125	50



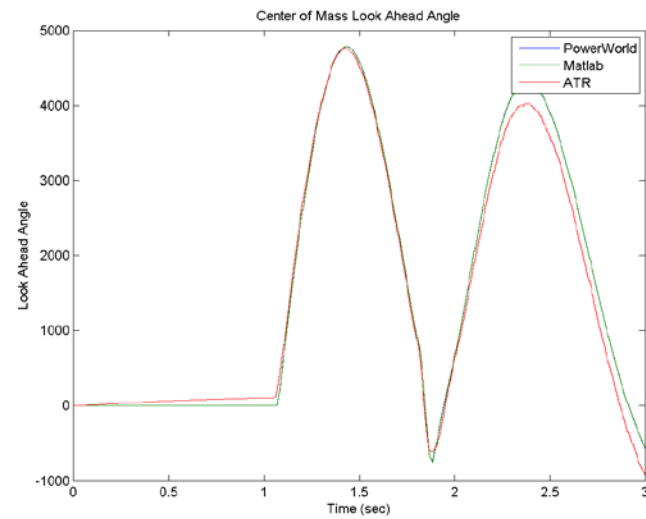
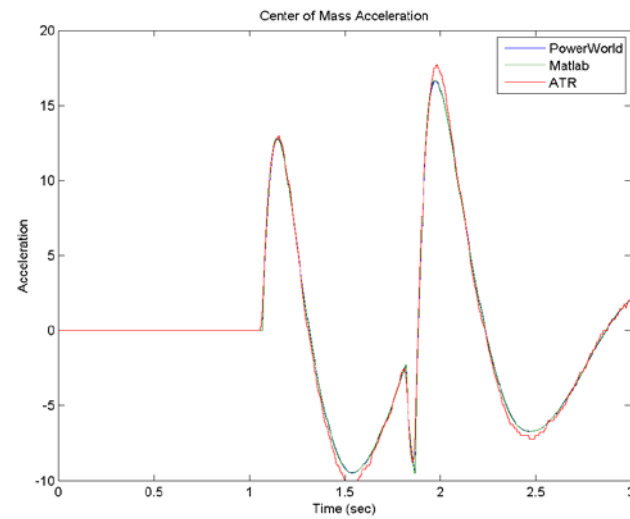
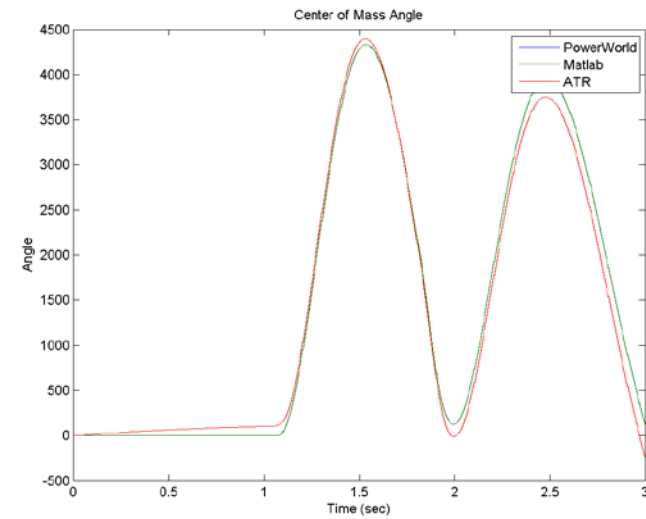
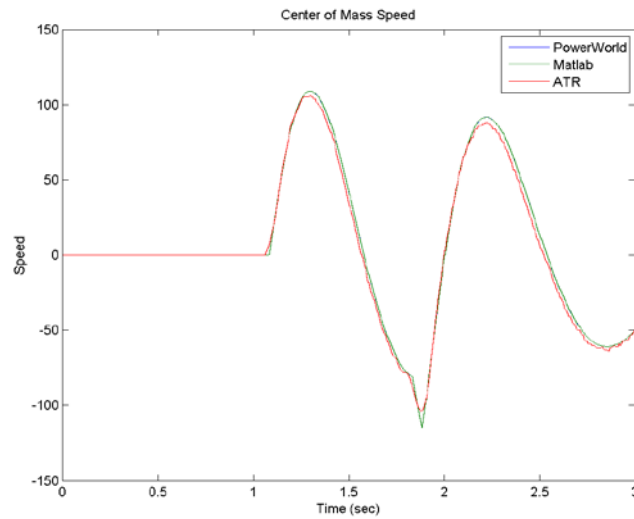
15ls1a_1bvgr12_ur

	Unit 1 Trip Time (sec)	Unit 2 Trip Time (sec)	Unit 3 Trip Time (sec)	Unit 4 Trip Time (sec)	Final MW Tripped	Final % Tripped
ATR	1.2499	0	1.2499	2.0331	1930	85
Matlab	1.2465	0	1.2465	2.0398	1930	85
PowerWorld	1.2465	0	1.2465	2.0398	1930	85



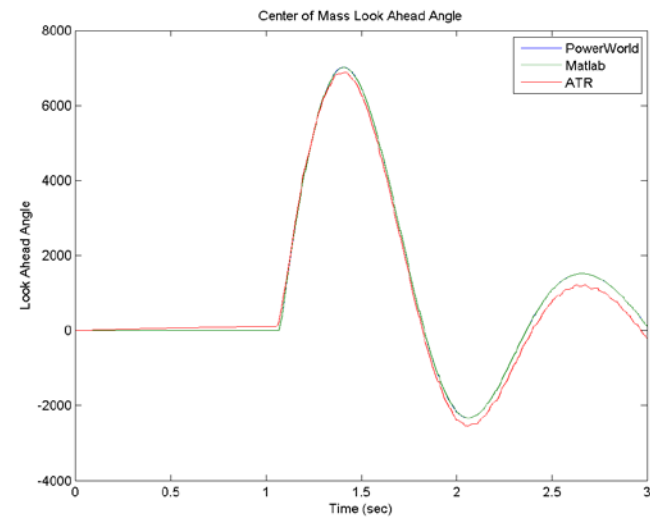
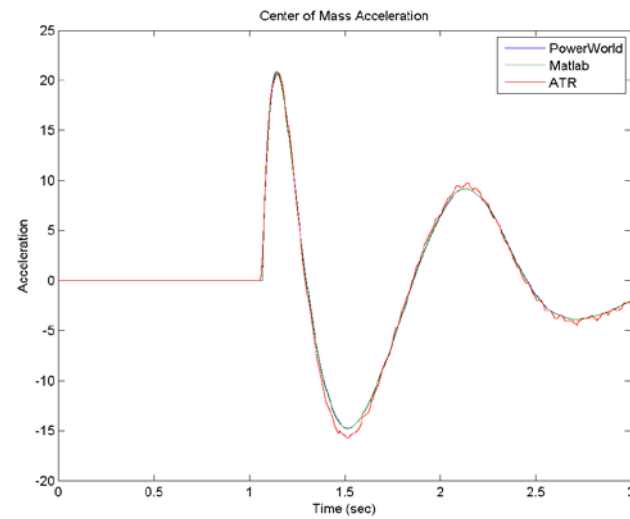
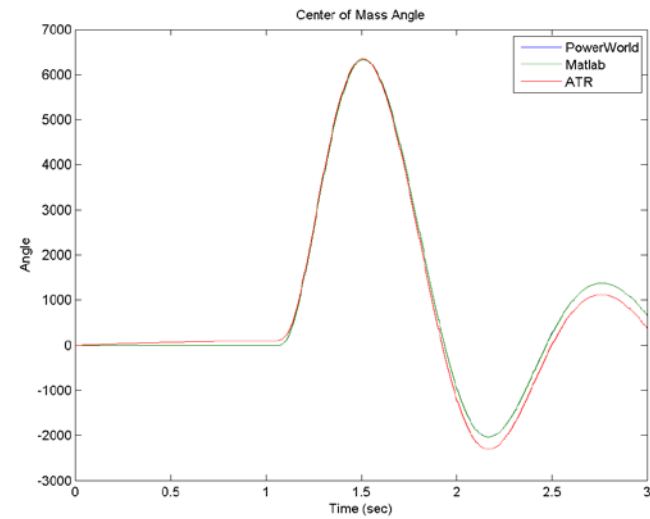
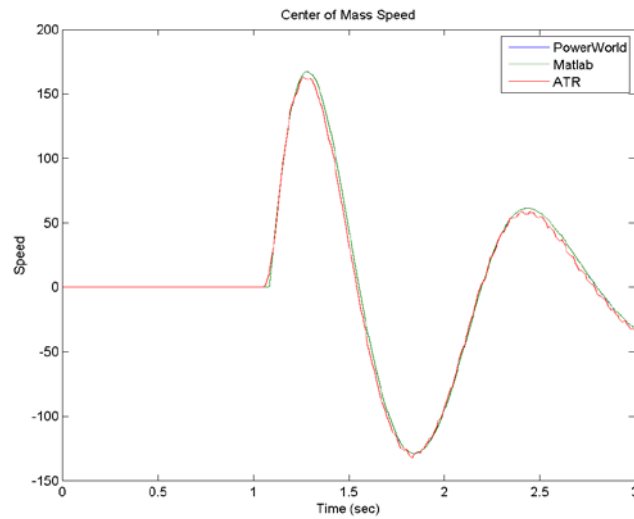
15ls1a_1bvgr1_ur

	Unit 1 Trip Time (sec)	Unit 2 Trip Time (sec)	Unit 3 Trip Time (sec)	Unit 4 Trip Time (sec)	Final MW Tripped	Final % Tripped
ATR	0	0	0	0	0	0
Matlab	0	0	0	0	0	0
PowerWorld	0	0	0	0	0	0



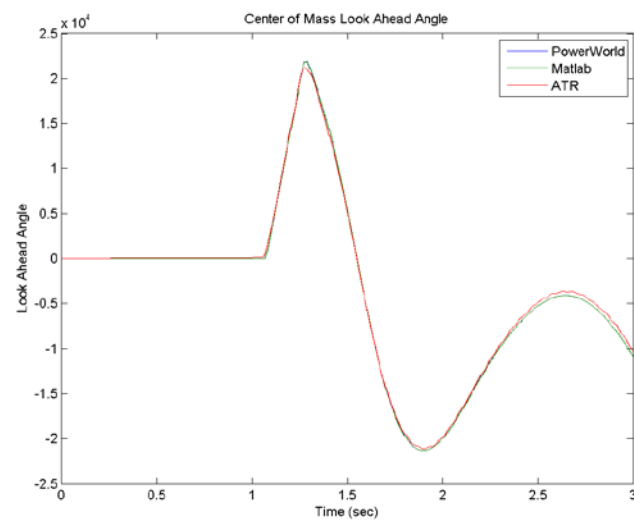
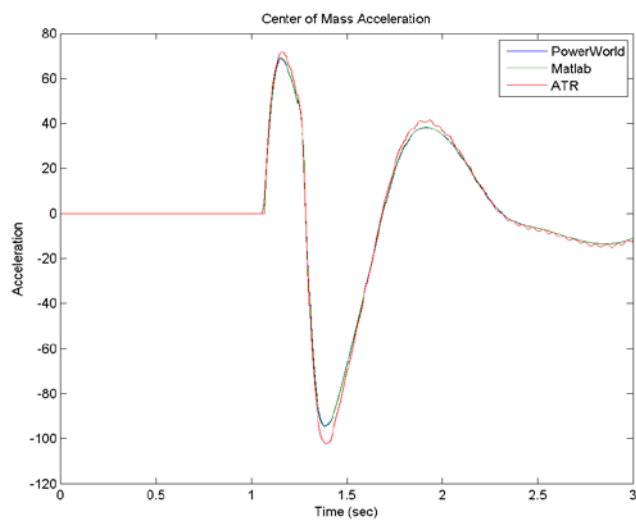
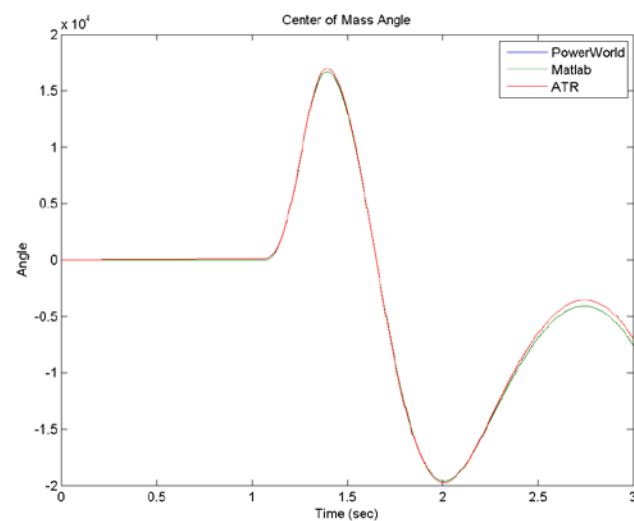
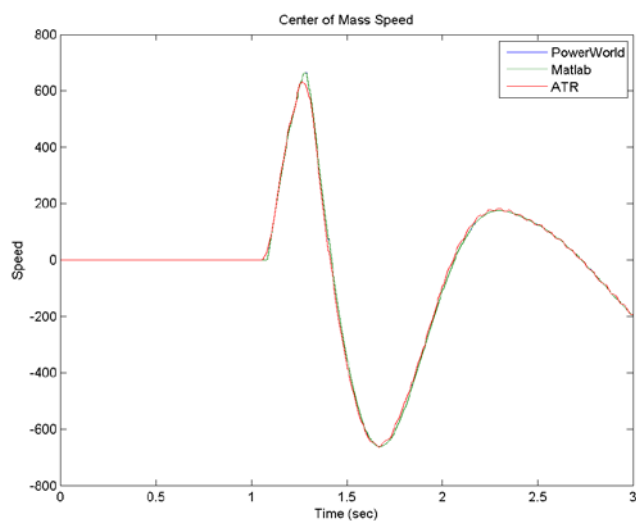
15ls1a_1csbv1

	Unit 1 Trip Time (sec)	Unit 2 Trip Time (sec)	Unit 3 Trip Time (sec)	Unit 4 Trip Time (sec)	Final MW Tripped	Final % Tripped
ATR	0	0	0	0	0	0
Matlab	0	0	0	0	0	0
PowerWorld	0	0	0	0	0	0



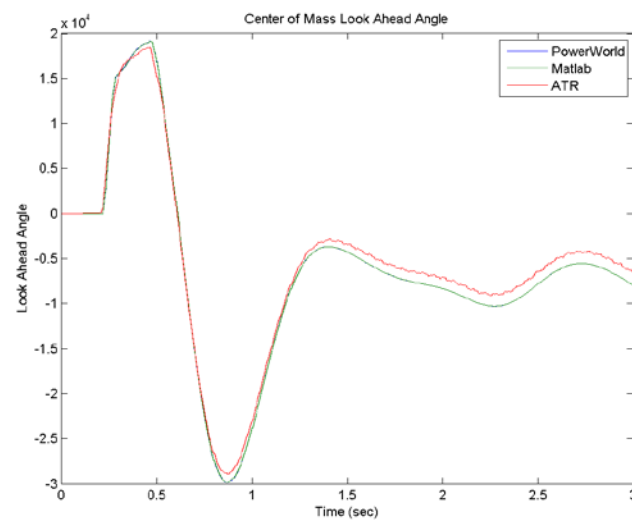
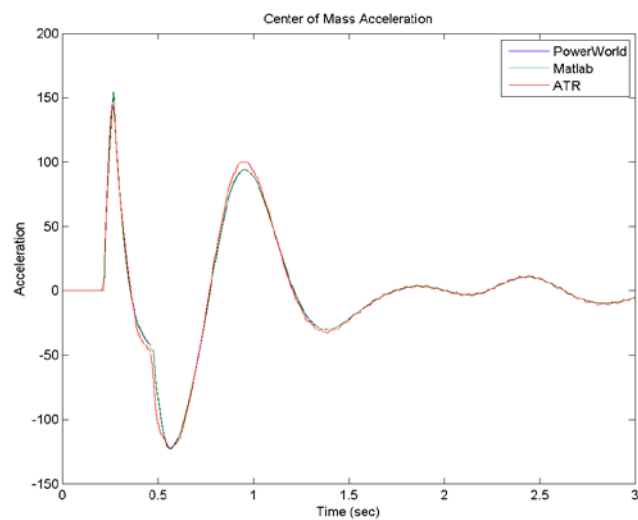
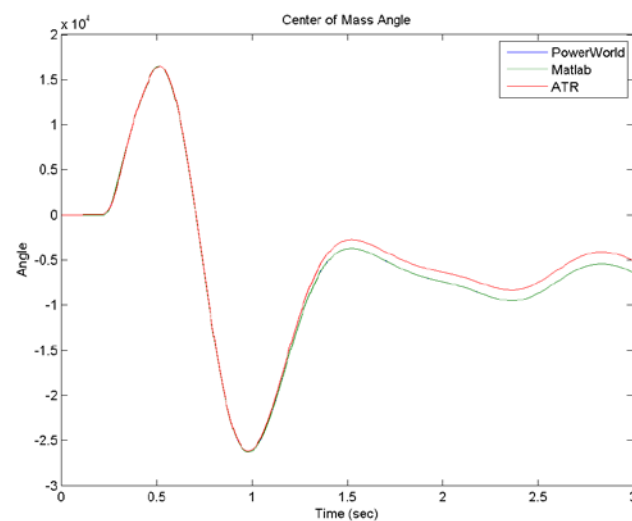
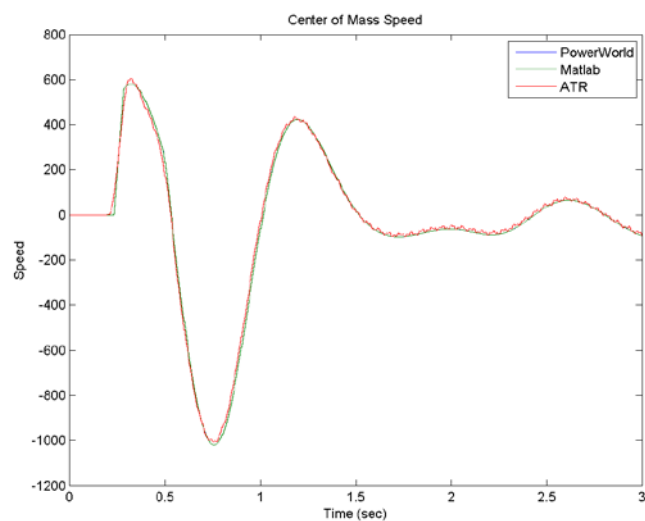
15ls1a_1grtt12

	Unit 1 Trip Time (sec)	Unit 2 Trip Time (sec)	Unit 3 Trip Time (sec)	Unit 4 Trip Time (sec)	Final MW Tripped	Final % Tripped
ATR	1.1899	0	1.1899	0	1125	50
Matlab	1.1899	0	1.1899	0	1125	50
PowerWorld	1.1899	0	1.1899	0	1125	50



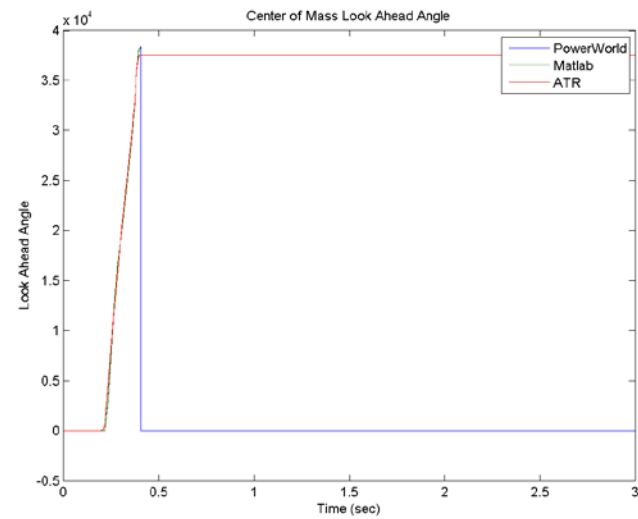
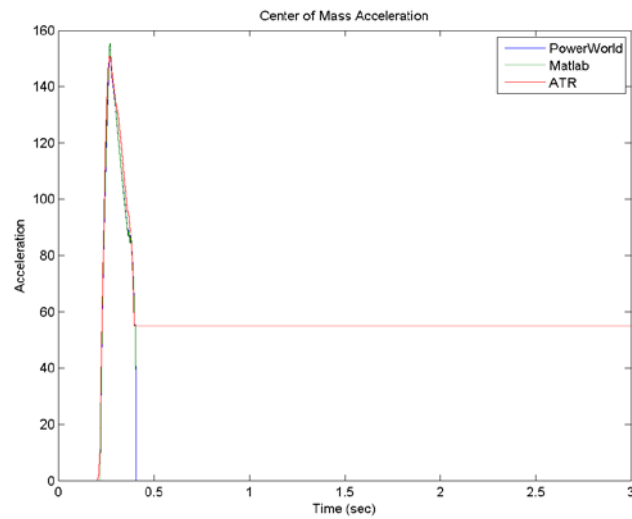
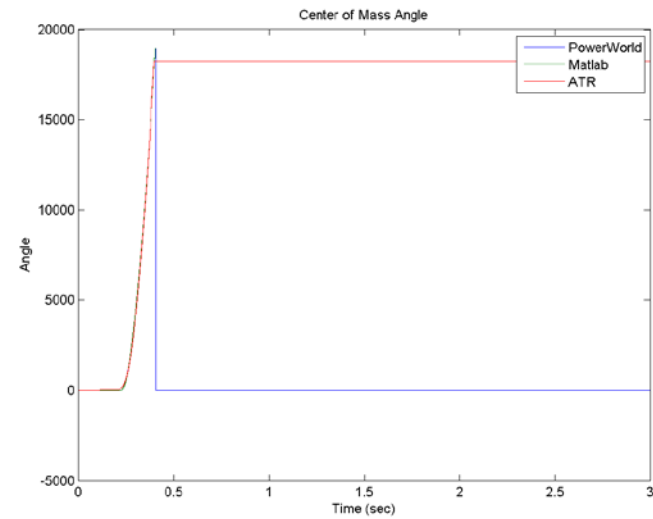
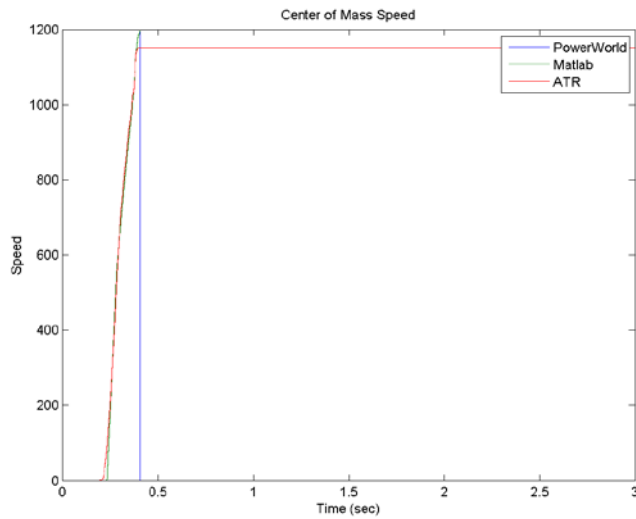
15ls1a_3bvgr1

	Unit 1 Trip Time (sec)	Unit 2 Trip Time (sec)	Unit 3 Trip Time (sec)	Unit 4 Trip Time (sec)	Final MW Tripped	Final % Tripped
ATR	0	0	0	0	0	0
Matlab	0	0	0.39329	0	795	35
PowerWorld	0	0	0.39329	0	795	35



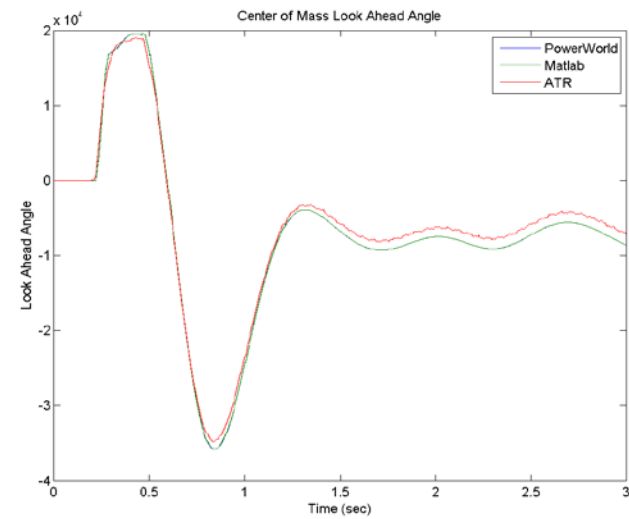
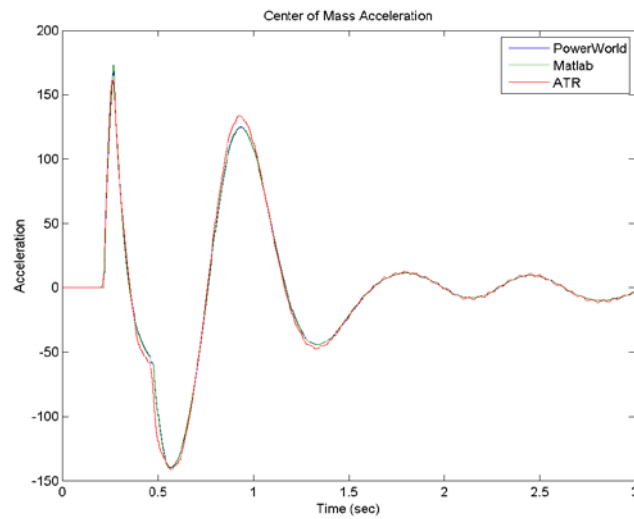
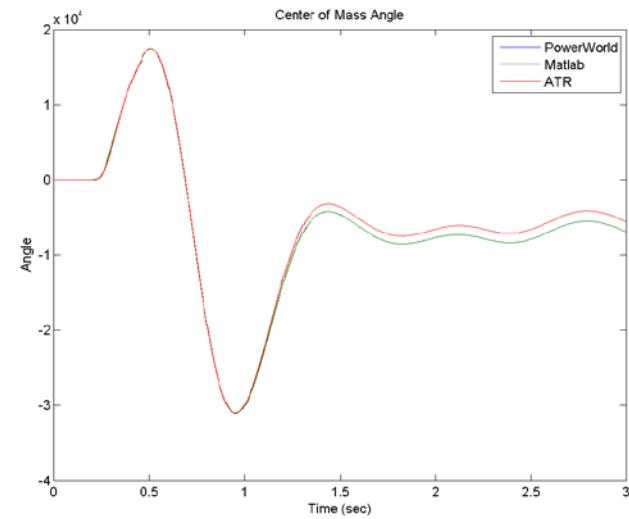
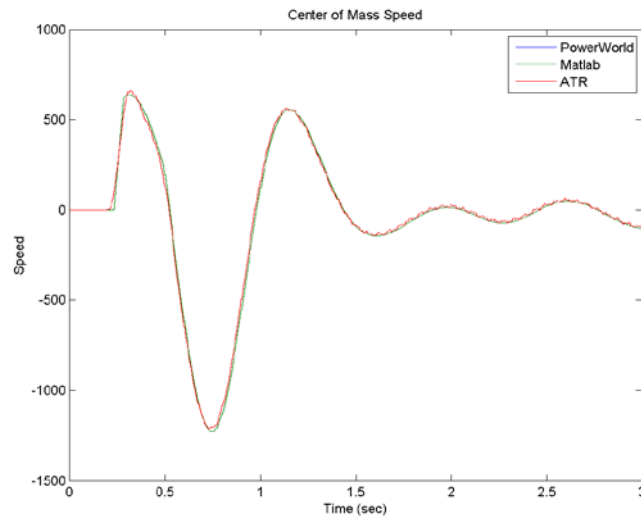
15ls1a_3bvgr12

	Unit 1 Trip Time (sec)	Unit 2 Trip Time (sec)	Unit 3 Trip Time (sec)	Unit 4 Trip Time (sec)	Final MW Tripped	Final % Tripped
ATR	0.30664	0	0.30664	0.3233	1930	85
Matlab	0.3133	0.32663	0.3133	0.38663	2260	100
PowerWorld	0.3133	0.32663	0.3133	0.38663	2260	100



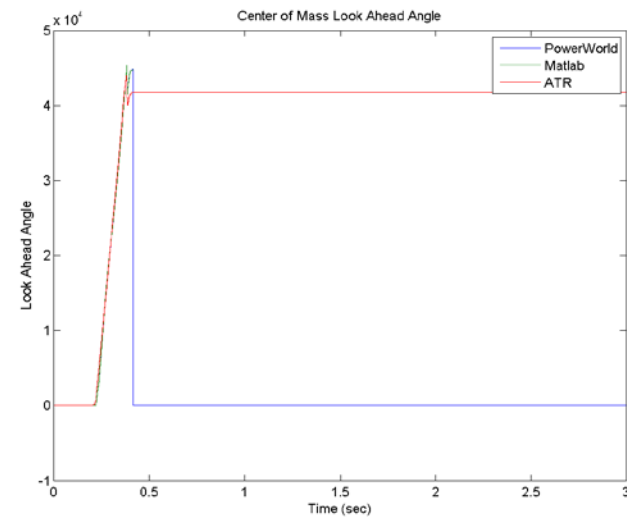
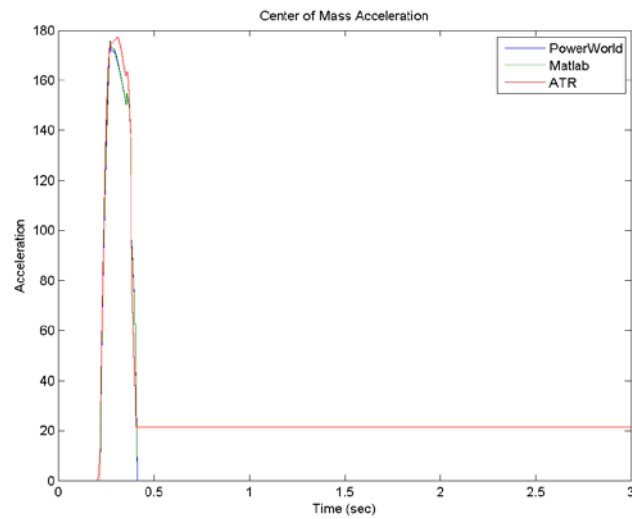
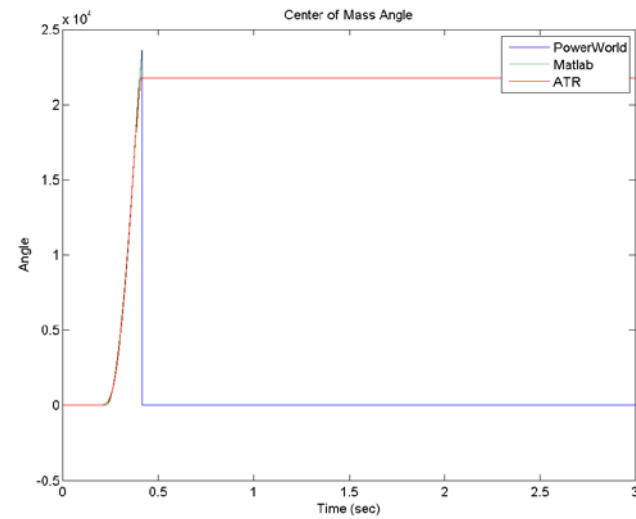
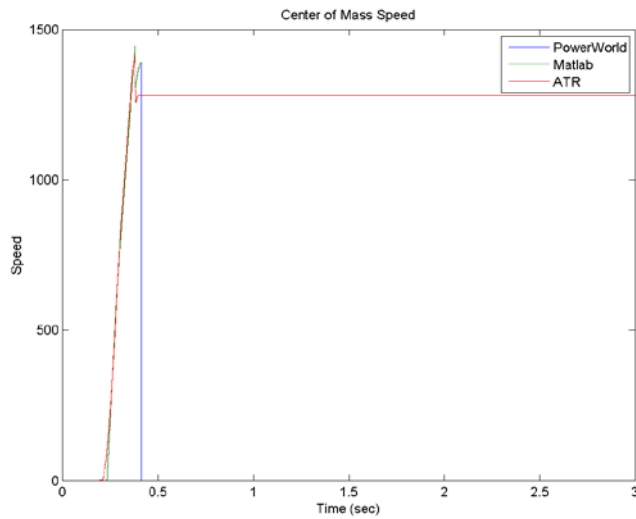
15ls1a_3csbv1

	Unit 1 Trip Time (sec)	Unit 2 Trip Time (sec)	Unit 3 Trip Time (sec)	Unit 4 Trip Time (sec)	Final MW Tripped	Final % Tripped
ATR	0	0	0	0	0	0
Matlab	0	0	0.39329	0	795	35
PowerWorld	0	0	0.39329	0	795	35



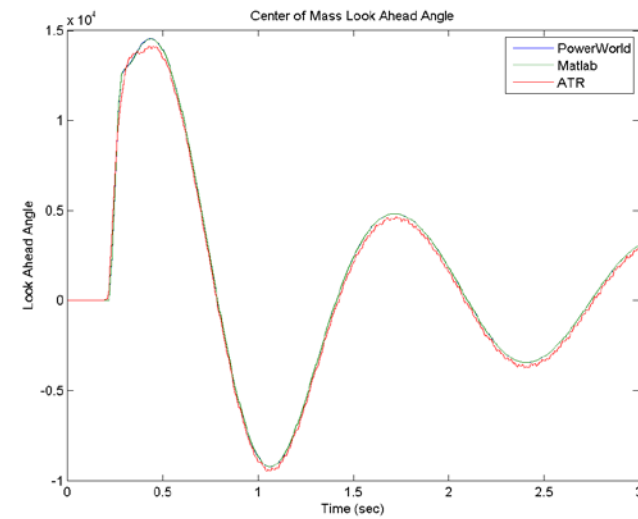
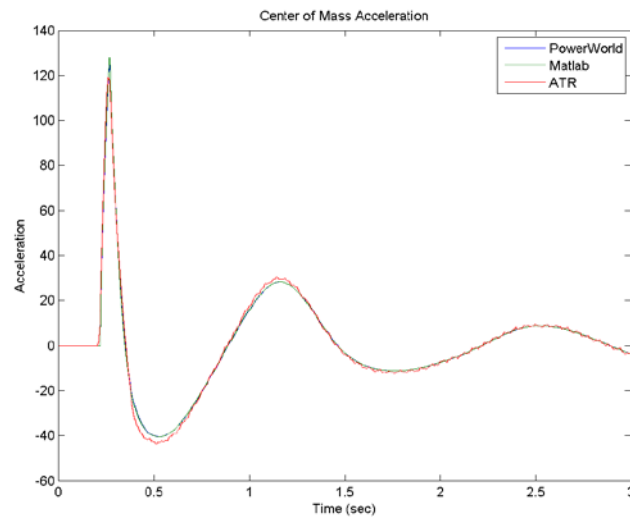
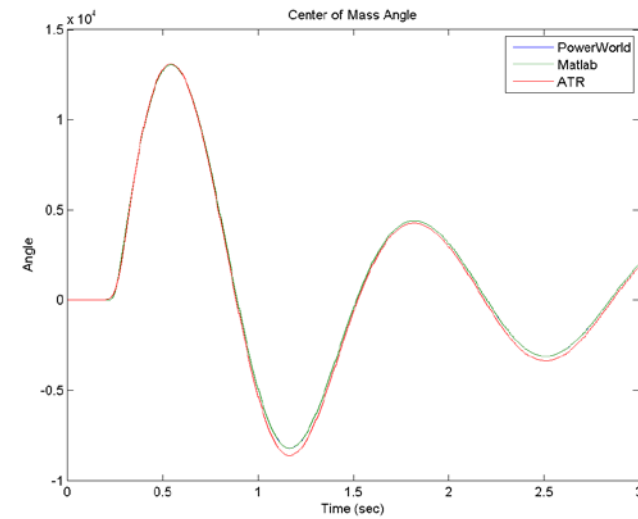
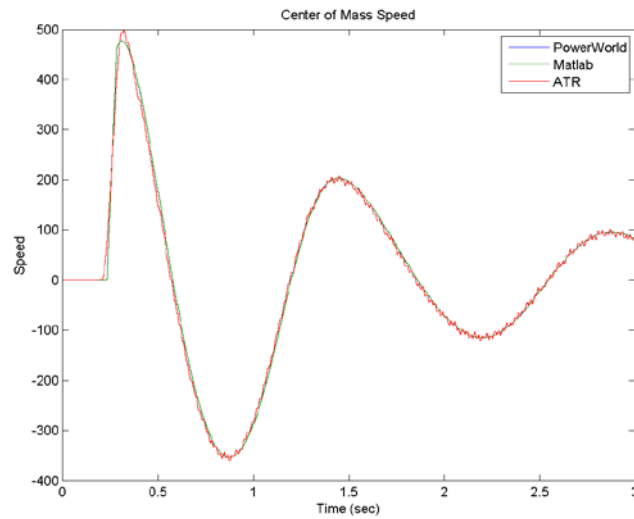
15ls1a_3csbv12

	Unit 1 Trip Time (sec)	Unit 2 Trip Time (sec)	Unit 3 Trip Time (sec)	Unit 4 Trip Time (sec)	Final MW Tripped	Final % Tripped
ATR	0.3033	0.36996	0.3033	0.30664	2260	100
Matlab	0.30664	0.36996	0.30664	0.3133	2260	100
PowerWorld	0.3033	0.36996	0.3033	0.3133	2260	100



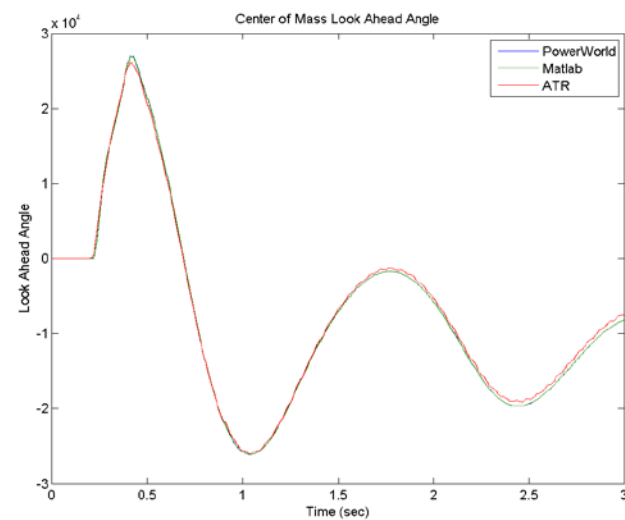
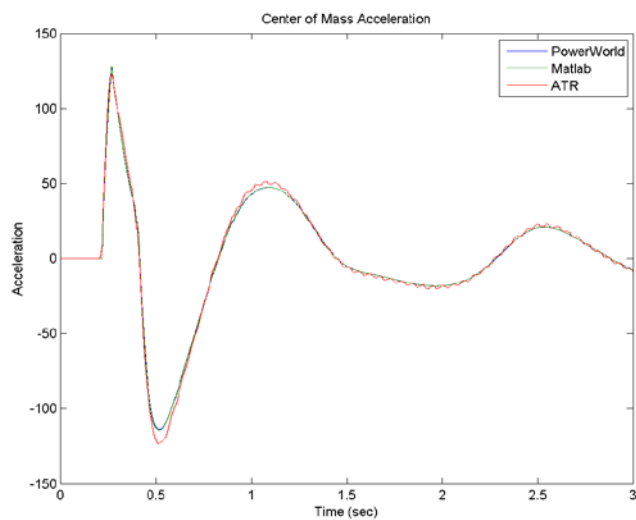
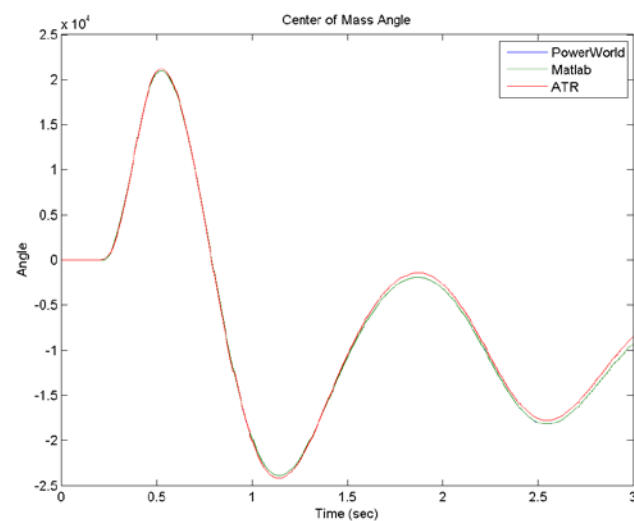
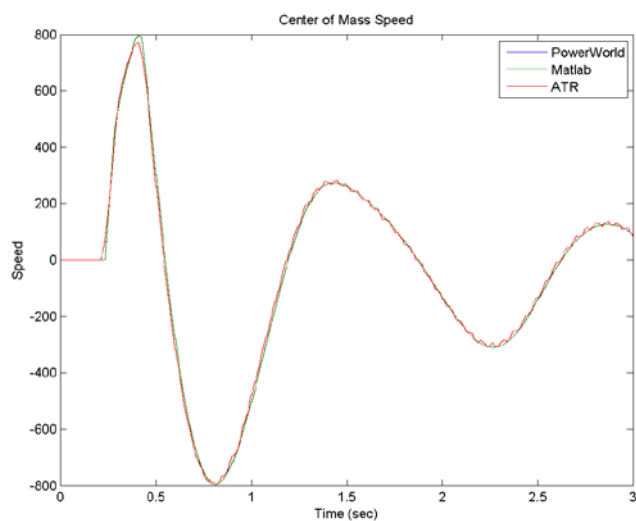
15ls1a_3grtt1

	Unit 1 Trip Time (sec)	Unit 2 Trip Time (sec)	Unit 3 Trip Time (sec)	Unit 4 Trip Time (sec)	Final MW Tripped	Final % Tripped
ATR	0	0	0	0	0	0
Matlab	0	0	0	0	0	0
PowerWorld	0	0	0	0	0	0



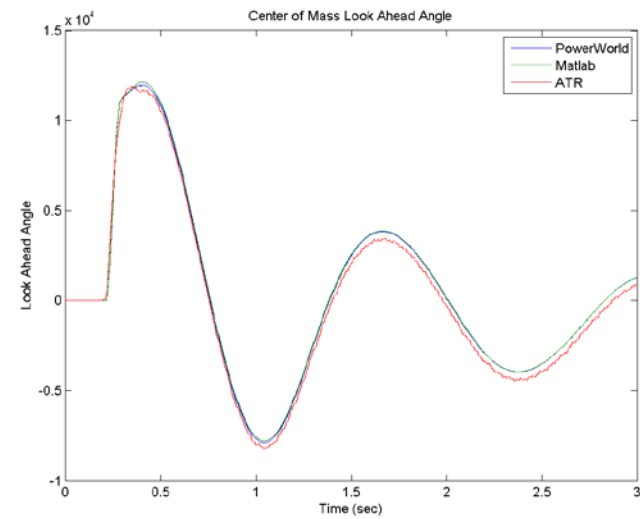
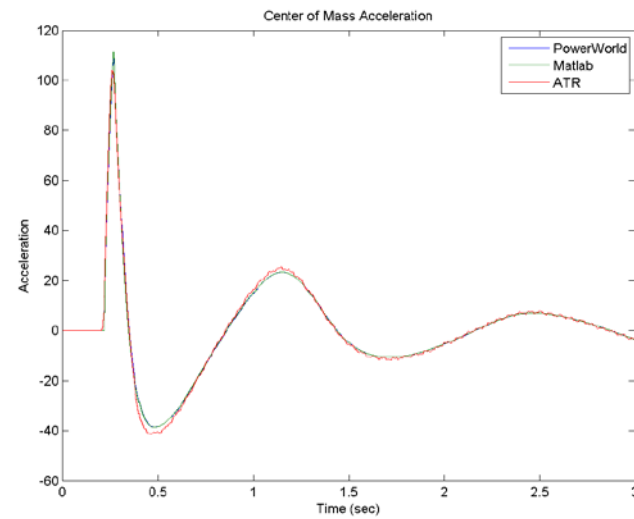
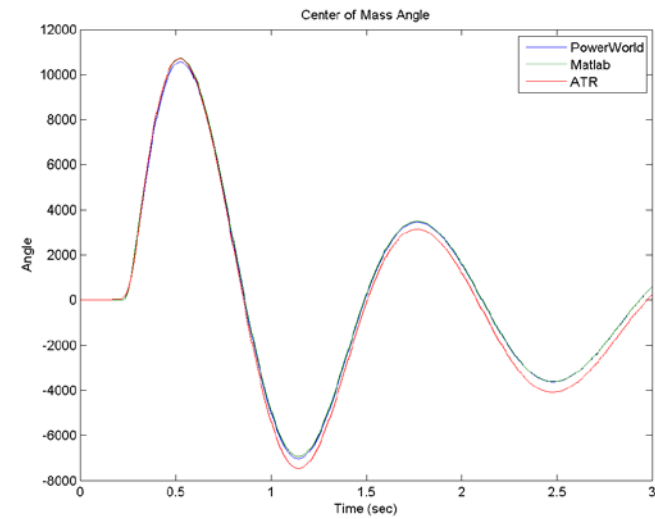
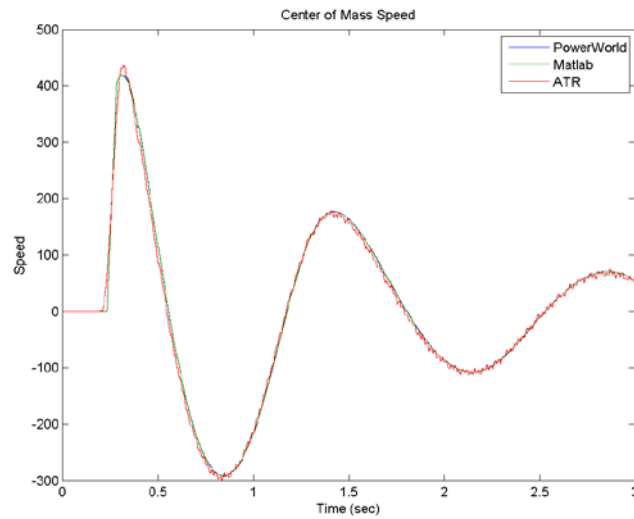
15ls1a_3grtt12

	Unit 1 Trip Time (sec)	Unit 2 Trip Time (sec)	Unit 3 Trip Time (sec)	Unit 4 Trip Time (sec)	Final MW Tripped	Final % Tripped
ATR	0.31997	0	0.31997	0	1125	50
Matlab	0.32997	0	0.32997	0	1125	50
PowerWorld	0.32997	0	0.32997	0	1125	50



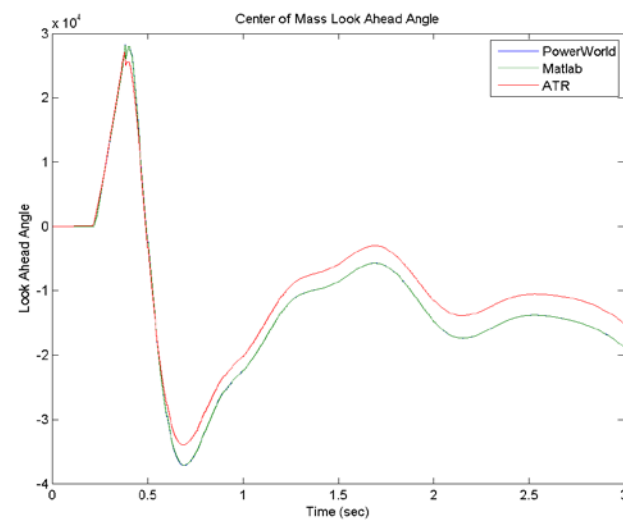
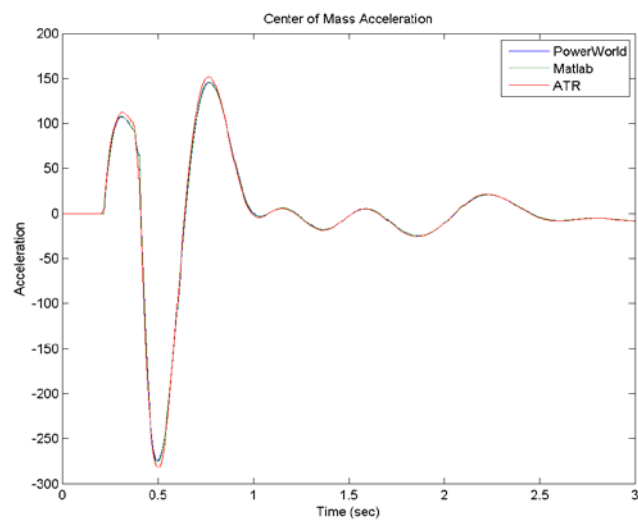
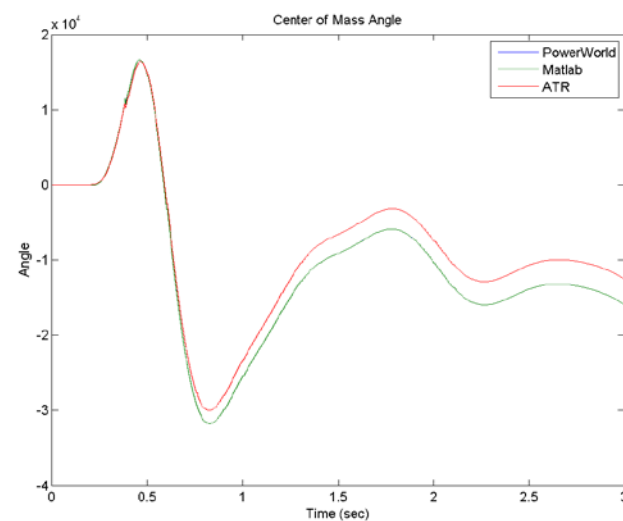
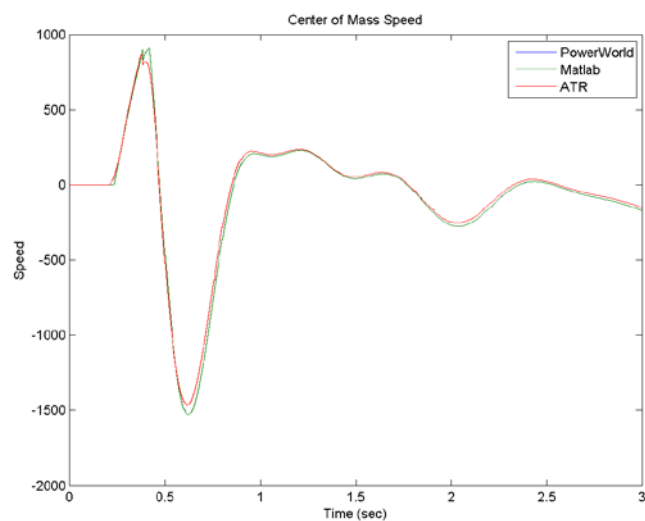
15ls1a_3ttbb

	Unit 1 Trip Time (sec)	Unit 2 Trip Time (sec)	Unit 3 Trip Time (sec)	Unit 4 Trip Time (sec)	Final MW Tripped	Final % Tripped
ATR	0	0	0	0	0	0
Matlab	0	0	0	0	0	0
PowerWorld	0	0	0	0	0	0



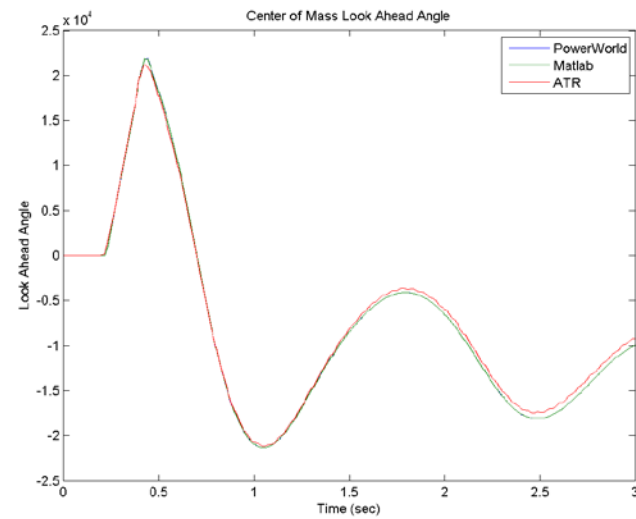
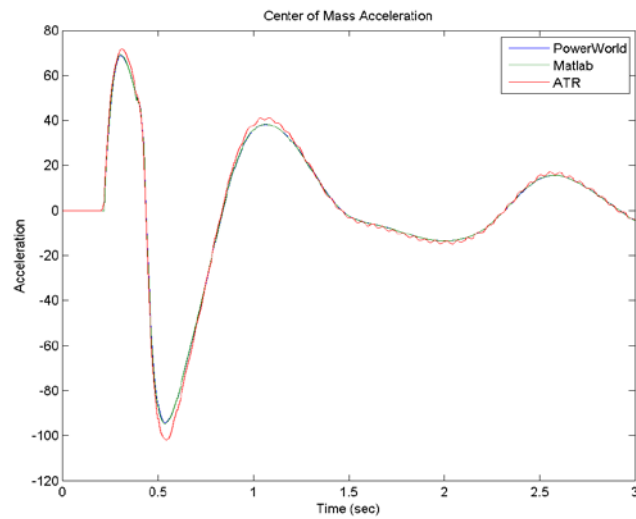
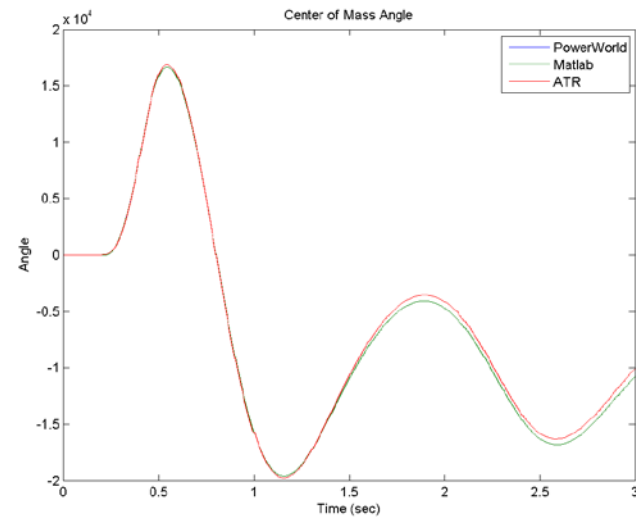
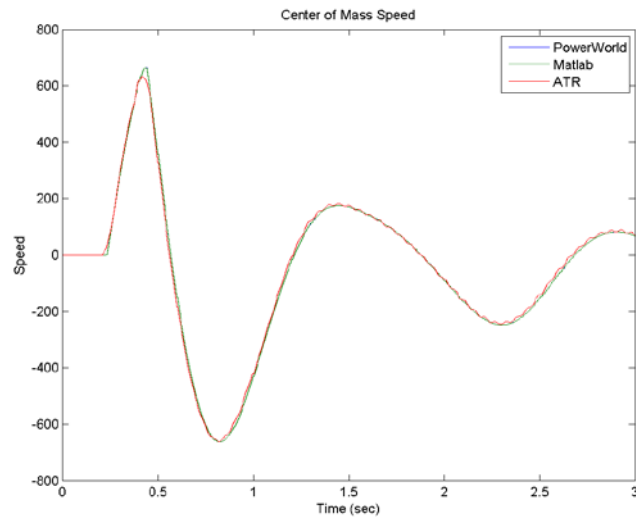
15ls1a_bvgr12

	Unit 1 Trip Time (sec)	Unit 2 Trip Time (sec)	Unit 3 Trip Time (sec)	Unit 4 Trip Time (sec)	Final MW Tripped	Final % Tripped
ATR	0.32663	0	0.31997	0.31997	1930	85
Matlab	0.32997	0	0.31664	0.31664	1930	85
PowerWorld	0.32997	0	0.31664	0.31664	1930	85



15ls1a_grtt12

	Unit 1 Trip Time (sec)	Unit 2 Trip Time (sec)	Unit 3 Trip Time (sec)	Unit 4 Trip Time (sec)	Final MW Tripped	Final % Tripped
ATR	0.3433	0	0.3433	0	1125	50
Matlab	0.33997	0	0.33997	0	1125	50
PowerWorld	0.33997	0	0.33997	0	1125	50





Appendix D PowerWorld Pseudo Code

The following PowerWorld code contains the function used for the algorithms and the generator relay trip.

Initialization:

```
PeakToPeakDetection := 0;
HighPeakDetected := false;
LowPeakDetected := false;
NeedToCalculateNewPeaktoPeakAmplitude := true;

AllowSecondPeakDetectionTrippingWhenSecondPickisFound := false;
Global_SPDPeak_High := 0;
Global_SPDPeak_Low := 0;

TDAC_DoAlgorithms := True;
TDACtimer_Initialized := false;
TDACtimer_Set := false;
BlockedANAC := false;
BlockedTDAC10 := false;
BlockedTDAC8 := false;
BlockedTDAC68 := false;
BlockedTDAC6 := false;
BlockedTDAC52 := false;
BlockedOACL := false;
BlockedACSP := false;
BlockedOSPD := false;
BlockedACAN := false;
BlockedOANG := false;
BlockedSPAN := false;

speed_up := false;
speed_reset := false;
angle_up := false;
angle_reset := false;

fpeakspd_ETS := 0;
fpeakacl_ETS := 0;
GrossEnable_Time := 0;
```

Following code is called at the end of each time-step

//List of Local Variables

```
StoringMaxTrippctMWValue : single;
FirstPeakPercentValue : single;
PeaktoPeakAmplitude : single;
TripMessage : string;
Local_CurrentTrippedAmount : single;
DynOcsTrip : boolean;
local_Event : TTxEvtObject;
IndUnitCt : integer;
theGenATR : TTxGenOther_ATRRELAY_Generic_PWGenRelayModel;
LocalTotal_Available : single;
local_TripPct : double; // jamie
local_BlockedOtherAlgorithms : boolean;
tempBoo : boolean;
localReset : boolean;
i : integer;
theBus : TTxBusObject;
LoadCountIntatBus, LoadCountIntAtFeeder, LoadCountTotal : integer;
TheLoad : TTxLoadObject;
```



Local Function here

```
procedure LOCAL_Process(theTripPct : double; theTripMessage : string;
                        var theActive, theOtherBlock : boolean);
begin
  TripMessage := theTripMessage;
  local_TripPct := theTripPct;
  theActive := true;
  theOtherBlock := true;
end;
```

Actual Code called at the end of each time-step

```
// Calculate the Center of Mass just weighted averages of block diagram outputs
UpdateCenterOfMassValues;

// -----//
// Previous4cyclesIndex shows the index of the present value in the Previous4cyclesPoints Array.
// Previous4cyclesPoints is a 1-based array. So just do a mod with the Count.
// If we're "at the end" then Previous4cyclesPoints = Count so MOD will be 0 and we'll wrap
// -----//
// Advance forward in "wrapping" array of previous points!
Previous4cyclesIndex := (Previous4cyclesIndex mod Previous4cyclesPoints.Count) + 1;

Previous4cyclesPoints.fVector[Previous4cyclesIndex] := fcmac1;
localReset := false;
// Determine if angle above noise threshold
if GrossEnable then begin
  if (fcmang > 1500) then begin
    angle_up := true;
    angle_reset := false;
  end;
  // Determine if angle below noise threshold following an up
  if (fcmang < -1500) and (angle_up) then begin
    angle_reset := true;
  end;
  // Determine if speed above noise threshold
  if (fcmspd > 50) then begin
    speed_up := true;
    speed_reset := false;
  end;
  // Determine if speed below noise threshold following an up
  if (fcmspd < -50) and (speed_up) then begin
    speed_reset := true;
  end;
  //if (fcmang <= -1500) and (fcmspd <= -50) then begin
  if (angle_reset) and (speed_reset) then begin
    localReset := true; // Reset the peak values
    GrossEnable := false;
    Report Event: ' (Detected cmang < -1500 and cmspd < -50, Algorithms Disabled)';
  end;
  // 2.5 Seconds (Hard Coded)
  if (GrossEnable and ((SimTime - GrossEnable_Time) >= 2.5)) then begin
    localReset := true; // Reset the peak values
    GrossEnable := false;
    Report Event: ' (Timer Limit Hit. Enble is set to false and Reset is set to true)';
  end;
end; // if GrossEnable

if localReset then begin
  // Reset all of the algorithms every time Enable is set and rest is true
  PeakToPeakDetection := 0;
  HighPeakDetected := false;
  LowPeakDetected := false;
  NeedToCalculateNewPeaktoPeakAmplitude := true;

  AllowSecondPeakDetectionTrippingWhenSecondPickisFound := false;
  Global_SPDPeak_High := 0;
```



```
Global_SPDPeak_Low := 0;

TDAC_DoAlgorithms := True;
TDActimer_Initialized := false;
TDActimer_Set := false;
BlockedANAC := false;
BlockedTDAC10 := false;
BlockedTDAC8 := false;
BlockedTDAC68 := false;
BlockedTDAC6 := false;
BlockedTDAC52 := false;
BlockedOACL := false;
BlockedACSP := false;
BlockedOSPD := false;
BlockedACAN := false;
BlockedOANG := false;
BlockedSPAN := false;

speed_up := false;
speed_reset := false;
angle_up := false;
angle_reset := false;

fpeakspd_ETS := 0;
fpeakacl_ETS := 0;
Report Event: ' (Peak Values Reset)';
end
else begin
  if fcmspd > fpeakspd_ETS then fpeakspd_ETS := fcmspd;
  if fcmacl > fpeakacl_ETS then begin
    fpeakacl_ETS := fcmacl;
    BlockedANAC := false; // "ANAC is reset after a new maximum acceleration is detected.
  end;
end;
if (fcmacl >= 20) and (not GrossEnable) then begin
  // Set GrossEnable to true. Used in Resetting the Peak value and will enable the "Algorithms"
  GrossEnable := true;
  GrossEnable_Time := IntegrationTime;
  Report Event: ' (Detected cmacl > 20, Algorithms Enabled)';
end;
// -----ALGORITHMS -----
if GrossEnable then begin
  local_TripPct := 0;
  DynOcsTrip := false;
  TripMessage := '';

  //-----
  // Dynamic Oscillation -----
  // Note: When dynamic oscillations become larger than the normal noise, positive and negative
  // values are subtracted to compute the peak-to-peak amplitude and successive values of the
  // amplitude are compared.
  // The following must all be true for algorithm to issue a trip:
  // • Peak-to-peak amplitude must increase for the last 3 consecutive cycles
  // • The amplitude must be greater than 200 counts
  // • The previous minimum peak must be less than -165 counts
  // • The consecutive peaks must have occurred within the last 2 seconds from each other
  // If all are true, a trip call of 14 percent will be made.
  // Similar to the Second Peak Over-speed algorithm, a peak is not considered a peak unless
  // it has reached a value of ±100 counts. The peak is also not detected until it has gone up
  // or down by either 20 percent of the peak value, or 12 counts, whichever is greater.
  local_BlockedOtherAlgorithms := false; // Reset this at every timestep

  //-----
  // Storing Local High and Low Speed values and also the Global High and Low Speed value used
  // in the Second Peak Over-Speed and Dynamic Oscillation Algorithms Values needed
  //-----

  if fcmspd > 0 then begin
    if fcmspd >= 100 then begin
      if fcmspd > Global_SPDPeak_High then begin
```



```
Global_SPDPeak_High := fcmspd;
// For Dynamic Oscillation Something Change then needs to
// restart time of peak to peak check
NeedToCalculateNewPeaktoPeakAmplitude := true;
end;
end;
// Check for High Peak Detection
if (Global_SPDPeak_High <> 0) then begin
    HighPeakDetected := (fcmspd*PWInvert(Global_SPDPeak_High) <= 0.80) and
                        ((Global_SPDPeak_High - fcmspd) >= 12);
end;
end;
else if fcmspd < 0 then begin
    if fcmspd <= -100 then begin
        if fcmspd < Global_SPDPeak_Low then begin
            Global_SPDPeak_Low := fcmspd;
            // For Dynamic Oscillation Something Change then needs to
            // restart time of peak to peak check
            NeedToCalculateNewPeaktoPeakAmplitude := true;
        end;
    end;
    // Check for Low Peak Detection
    if (Global_SPDPeak_Low <> 0) then begin
        LowPeakDetected := (-fcmspd*PWInvert(Global_SPDPeak_Low) >= 0.80) and
                        (abs(Global_SPDPeak_Low - fcmspd) >= 12);
    end;
end;
end;
//-----//
//-- Dynamic Oscillation -----//
//-----//
// Only do this once when just changed is false
if HighPeakDetected and LowPeakDetected and NeedToCalculateNewPeaktoPeakAmplitude then begin
    NeedToCalculateNewPeaktoPeakAmplitude := false;
    PeaktoPeakAmplitude := Global_SPDPeak_High - Global_SPDPeak_Low;
    if (PeakToPeakDetection > 0) then begin
        // Reset the time because the Amplitude is not increasing
        if PeaktoPeakAmplitude < Old_PeaktoPeakAmplitude then begin
            PeakToPeakDetection := 0;
        end
    end
    else begin
        Old_PeaktoPeakAmplitude := PeaktoPeakAmplitude;
        if PeakToPeakDetection = 1 then Second_PeaktoPeakTime := fSimTime; // Stored 2nd Time
    end;
end;
// Check for case of more than 3 consecutives amplitudes increases
DynOcsTrip := false;
if PeakToPeakDetection >= 2 then begin

    Third_PeaktoPeakTime := fSimTime;
    if (TPWTimeStamp_DiffSeconds(First_PeaktoPeakTime,Third_PeaktoPeakTime) > 2) then begin
        // Reset Everything again
        PeakToPeakDetection := 0;
    end
    else begin
        if Global_SPDPeak_Low < -165 then begin
            local_TripPct := 0.14;
            TripMessage := 'Dynamic Oscillation';
            DynOcsTrip := true;
        end
        else begin
            Old_PeaktoPeakAmplitude := PeaktoPeakAmplitude;
            // Set the first time to the Second time (3 consecutives increasing
            // have to be within 2 seconds)
            First_PeaktoPeakTime := Second_PeaktoPeakTime;
            Second_PeaktoPeakTime := fSimTime;
        end;
    end;
end;
// if PeakToPeakDetection >= 2 then begin
if PeakToPeakDetection = 0 then begin // For Resetting and Initial simulation
    First_PeaktoPeakTime := fSimTime;
```



```
Old_PeaktoPeakAmplitude := PeaktoPeakAmplitude;
end;
PeakToPeakDetection := PeakToPeakDetection + 1
end; // if HighPeakDetected and LowPeakDetected and NeedToCalculateNewPeaktoPeakAmplitude
-----//
// Second Peak Over-Speed -----//
// Note: When a positive local maximum speed greater than normal noise is detected, the value
// is stored. Until the speed goes negative, subsequent values of speed are compared to the
// previous peak.
// If a new value exceeds the old peak, this algorithm calls for a trip.
// The amount to be tripped depends on the magnitude of the first peak.
// If the first peak is 450 counts or greater, the trip amount is 81 percent.
// If the first peak is 175 or greater, but less than 450, the trip amount is 48 percent.
// If the initial peak is less than 175, the algorithm will not issue a trip.
// A peak is not considered a peak unless it has reached a value of 100 counts.
// The peak is also not detected until it has gone down by either 20 percent of the peak
// value, or 12 counts (0.2 Seconds), whichever is greater. See Figure 5 for a
// representation of how this algorithm behaves.
if (fcmspd > 100) and (AllowSecondPeakDetectionTripped) then begin
  if fcmspd > Global_SPDPeak_Local_Detected then begin
    Global_SPDPeak_Local_Detected := fcmspd;
  end;
  if fcmspd < Global_SPDPeak_Local_Detected then begin
    FirstPeakPercentValue := (1-(fcmspd)*PWInvert(Global_SPDPeak_Local_Detected))*100;
    if (FirstPeakPercentValue <= 20) or
      ((Global_SPDPeak_Local_Detected - fcmspd) < 12)
    then begin
      FirstPeakTripDetection := false;
    end
  else begin
    Global_First_SPDPeak := Global_SPDPeak_Local_Detected;
    if not FirstPeakTripDetection then begin
      FirstPeakTripDetection := true;
      Report Event: ' (First Peak in Second Over-Speed Algorithm action detected.)';
    end;
    if FirstPeakTripDetection then begin
      if Global_First_SPDPeak > 450 then
        AllowSecondPeakDetectionTrippingWhenSecondPickisFound := true
      else if (Global_First_SPDPeak <= 450) and (Global_First_SPDPeak > 175) then
        AllowSecondPeakDetectionTrippingWhenSecondPickisFound := true
      else
        AllowSecondPeakDetectionTrippingWhenSecondPickisFound := false;
      end;
    end; // if fcmspd < Global_SPDPeak_Local_Detected
  end; // if fcmspd < Global_SPDPeak_Local_Detected

  if FirstPeakTripDetection then begin // it means it already found a first peak
    if Global_First_SPDPeak > Global_SPDPeak_Local_Detected then begin
      // it means it is a second Peak different from the First
      if AllowSecondPeakDetectionTrippingWhenSecondPickisFound then begin
        TripMessage := 'Second Peak Over-Speed';
        if Global_First_SPDPeak > 450 then
          local_TripPct := 0.81
        else if (Global_First_SPDPeak <= 450) and (Global_First_SPDPeak > 175) then
          local_TripPct := 0.48;
        AllowSecondPeakDetectionTripped := false;
      end // it means that the second peak now needs to be the first
    else begin
      FirstPeakTripDetection := false;
      Global_SPDPeak_Local_Detected := Global_First_SPDPeak;
    end;
  end;
end;
end;
end;
if fcmspd < 0 then begin
  FirstPeakTripDetection := false;
  Global_SPDPeak_Local_Detected := 0;
  Global_First_SPDPeak := 0;
end;
```



```
//-----//
// Time Triggered Acceleration (TDAC) -----//
If (TDAC_DoAlgorithms) Then Begin
  if ( fcmac1 - Previous4cyclesPoints.GetMinValue(i) ) >= 10 then begin
    // Acceleration has increased by at least 10 counts in the last
    // 4 electrical cycles then start timer
    if not TDACTimer_Set then begin
      TDACTimer_Set := true;
      TDACTimer := fSimTime;
      TDACTimer_Initialized := true; // AAA 03/16/15
      Report Event: ' (TDAC timer Pickup)';
    end;
  end
  else begin
    // Reset timer if necessary
    if TDACTimer_Set and (not TDACTimer_Initialized) then begin
      TDACTimer_Set := false;
      Report Event: ' (TDAC timer Reset)';
    end;
  end;
  if TDACTimer_Set and (TPWTimeStamp_DiffSeconds(fSimTime,TDACTimer) >= 0.2000) then begin
    if (fpeakac1_ETS >= 25) and (fcmang > 4000) then begin
      if (fcmac1 >= (70 - 0.500*fpeakac1_ETS)) then
        LOCAL_Process(0.48, 'TDAC 12.0', tempBoo, local_BlockedOtherAlgorithms)
      else if (fcmac1 >= (55 - 0.583*fpeakac1_ETS)) then
        LOCAL_Process(0.31, 'TDAC 12.0', tempBoo, local_BlockedOtherAlgorithms)
      else if (fcmac1 >= 10) then
        LOCAL_Process(0.31, 'TDAC 12.0', tempBoo, local_BlockedOtherAlgorithms);
    end;
    TDAC_DoAlgorithms := False; // To eliminate possibility of activating algorithms again
  end;
  if TDACTimer_Set and (TPWTimeStamp_DiffSeconds(fSimTime,TDACTimer) >= 0.1667) then begin
    if (fpeakac1_ETS >= 25) and (fcmang > 4000) and (not BlockedTDAC10) then begin
      if (fcmac1 >= (80 - 0.525*fpeakac1_ETS)) then
        LOCAL_Process(0.48, 'TDAC 10.0', BlockedTDAC10, local_BlockedOtherAlgorithms)
      else if (fcmac1 >= (50 - 0.500*fpeakac1_ETS)) then
        LOCAL_Process(0.31, 'TDAC 10.0', BlockedTDAC10, local_BlockedOtherAlgorithms)
      else if (fcmac1 >= 13) then
        LOCAL_Process(0.31, 'TDAC 10.0', BlockedTDAC10, local_BlockedOtherAlgorithms);
      BlockedTDAC10 := true;
    end;
  end;
  if TDACTimer_Set and (TPWTimeStamp_DiffSeconds(fSimTime,TDACTimer) >= 0.1333) then begin
    if (fpeakac1_ETS >= 25) and (fcmang > 4000) and (not BlockedTDAC8) then begin
      if (fcmac1 >= (80 - 0.300*fpeakac1_ETS)) then
        LOCAL_Process(0.48, 'TDAC 8.0', BlockedTDAC8, local_BlockedOtherAlgorithms)
      else if (fcmac1 >= (25 - 0.050*fpeakac1_ETS)) then
        LOCAL_Process(0.31, 'TDAC 8.0', BlockedTDAC8, local_BlockedOtherAlgorithms);
      BlockedTDAC8 := true;
    end;
  end;
  if TDACTimer_Set and (TPWTimeStamp_DiffSeconds(fSimTime,TDACTimer) >= 0.1133) then begin
    if (fpeakac1_ETS >= 75) and (fcmang > 4000) and (not BlockedTDAC68) then begin
      if (fcmac1 >= (80 + 0.150*fpeakac1_ETS)) then
        LOCAL_Process(0.81, 'TDAC 6.8', BlockedTDAC68, local_BlockedOtherAlgorithms)
      else if (fcmac1 >= 83) then
        LOCAL_Process(0.64, 'TDAC 6.8', BlockedTDAC68, local_BlockedOtherAlgorithms)
      else if (fcmac1 >= 53) then
        LOCAL_Process(0.48, 'TDAC 6.8', BlockedTDAC68, local_BlockedOtherAlgorithms);
      BlockedTDAC68 := true;
    end;
  end;
  if TDACTimer_Set and (TPWTimeStamp_DiffSeconds(fSimTime,TDACTimer) >= 0.1) then begin
    if (fpeakac1_ETS >= 75) and (fcmang > 4000) and (not BlockedTDAC6) then begin
      if (fcmac1 >= (50 + 0.450*fpeakac1_ETS)) then
        LOCAL_Process(0.81, 'TDAC 6.0', BlockedTDAC6, local_BlockedOtherAlgorithms)
      else if (fcmac1 >= (60 + 0.300*fpeakac1_ETS)) then
        LOCAL_Process(0.64, 'TDAC 6.0', BlockedTDAC6, local_BlockedOtherAlgorithms)
      else if (fcmac1 >= 103) then
        then

```



```
        LOCAL_Process(0.64, 'TDAC 6.0', BlockedTDAC6, local_BlockedOtherAlgorithms)
    else if (fcmac1 >= (40.8 + 0.256*fpeakac1_ETS)) then
        LOCAL_Process(0.48, 'TDAC 6.0', BlockedTDAC6, local_BlockedOtherAlgorithms)
    else if (fcmac1 >= 85) then
        LOCAL_Process(0.48, 'TDAC 6.0', BlockedTDAC6, local_BlockedOtherAlgorithms);
    BlockedTDAC6 := true;
end;
end;
if TDAcTimer_Set and (TPWTimeStamp_DiffSeconds(fSimTime, TDAcTimer) >= 0.0867) then begin
    if (fpeakac1_ETS >= 25) and (fcmang > 4000) and (not BlockedTDAC52) then begin
        if (fcmac1 >= (115 + 0.225*fpeakac1_ETS)) then
            LOCAL_Process(0.81, 'TDAC 5.2', BlockedTDAC52, local_BlockedOtherAlgorithms)
        else if (fcmac1 >= (25 + 0.714*fpeakac1_ETS)) then
            LOCAL_Process(0.81, 'TDAC 5.2', BlockedTDAC52, local_BlockedOtherAlgorithms)
        else if (fcmac1 >= 148) then
            LOCAL_Process(0.81, 'TDAC 5.2', BlockedTDAC52, local_BlockedOtherAlgorithms);
        BlockedTDAC52 := true;
    end;
end;
end;
//-----//
// All these algorithms run in parallel thus only the highest trip is selected when more than
// one trip signal is detected. Some algorithms are blocked if they set to trip until Enable
// is set to false and reset the algorithm parameters.
//-----//
// Speed vs Look-Ahead (SPAN) -----//
if (fcmla6ang >= (12000 + 32.5*fpeakspd_ETS)) and
    (not local_BlockedOtherAlgorithms) and (not BlockedSPAN)
then begin
    LOCAL_Process(0.14, 'SPAN', BlockedSPAN, local_BlockedOtherAlgorithms)
end;
// Angle Triggered Acceleration (ANAC) -----//
if (fcmang >= 16000) and (not BlockedANAC) then begin
    if (fcmac1 >= (215 - 0.75*fpeakac1_ETS)) then
        LOCAL_Process(0.81, 'ANAC', BlockedANAC, local_BlockedOtherAlgorithms)
    else if (fcmac1 >= 46) then
        LOCAL_Process(0.81, 'ANAC', BlockedANAC, local_BlockedOtherAlgorithms)
    else if (fcmac1 >= (75.33 - 0.3167*fpeakac1_ETS)) then
        LOCAL_Process(0.64, 'ANAC', BlockedANAC, local_BlockedOtherAlgorithms)
    else if (fcmac1 >= -4) then
        LOCAL_Process(0.48, 'ANAC', BlockedANAC, local_BlockedOtherAlgorithms)
    else if (fcmac1 >= (20 - 0.30*fpeakac1_ETS)) then
        LOCAL_Process(0.48, 'ANAC', BlockedANAC, local_BlockedOtherAlgorithms)
    else if (fcmac1 >= (32 - 0.45*fpeakac1_ETS)) then
        LOCAL_Process(0.31, 'ANAC', BlockedANAC, local_BlockedOtherAlgorithms);
end;
// Over-Acceleration (OACL) -----//
if (fcmac1 >= 190) and (fcmang >= 4000) and
    (not local_BlockedOtherAlgorithms) and (not BlockedOACL)
then begin
    LOCAL_Process(0.81, 'OACL', BlockedOACL, local_BlockedOtherAlgorithms);
end;
// Acceleration vs Speed (ACSP) -----//
if (fcmspd >= (170 + 3.6*fpeakac1_ETS)) and (fcmang >= 4000) and
    (not local_BlockedOtherAlgorithms) and (not BlockedACSP)
then begin
    LOCAL_Process(0.48, 'ACSP', BlockedACSP, local_BlockedOtherAlgorithms);
end;
// Over-Speed (OSPD) -----//
if (fcmspd >= 840) and
    (not local_BlockedOtherAlgorithms) and (not BlockedOSPD)
then begin
    LOCAL_Process(0.48, 'OSPD', BlockedOSPD, local_BlockedOtherAlgorithms);
end;
// Acceleration vs Look-Ahead Angle (ACAN) -----//
if (fcmla6ang >= (12000 + 162.5*fpeakac1_ETS)) and
    (not local_BlockedOtherAlgorithms) and (not BlockedACAN)
then begin
    LOCAL_Process(0.14, 'ACAN', BlockedACAN, local_BlockedOtherAlgorithms);
```



```
end;
// Over-Angle (OANG) -----//
if (fcmla6ang >= 25000) and
   (not local_BlockedOtherAlgorithms) and (not BlockedOANG)
then begin
  LOCAL_Process(0.14, 'OANG', BlockedOANG, local_BlockedOtherAlgorithms);
end;

// Now find and trip units -----//
if TripMessage = 'Dynamic Oscillation' then DynOcsTrip := true;
if fAUnitHasTripped then begin
  // Start checking this only when a unit is already tripped, not from the simulation start
  // Note:
  // If a unit has tripped within the last half second
  // (or 2 seconds if trip request came from the Dynamic Oscillation algorithm)
  // the MW value (prevtripmw) is included in the totalmw amount.
  if (DynOcsTrip) and (TripMessage = 'Dynamic Oscillation') then begin
    if TPWTimeStamp_DiffSeconds(fSimTime, fAUnitHasTripped_Time) > 2.0 then begin
      // reset when the time since last trip is greater than 2.0 seconds
      fPreviousTrippedAmount := 0;
    end;
  end
  else begin
    if TPWTimeStamp_DiffSeconds(fSimTime, fAUnitHasTripped_Time) > 0.5 then begin
      // reset when the time since last trip is greater than 2.0 seconds
      fPreviousTrippedAmount := 0;
    end;
  end;
end;
end;

// Now call the function to find the tripping of units
LocalTotal_Available := 0;
for IndUnitCt := 1 to 4 do begin // AAA 03/02/15
  if (not GenATRRelays[IndUnitCt].MonitorForTrip) then begin
    LocalTotal_Available := LocalTotal_Available +
      GenATRRelays[IndUnitCt].PGenSystemSetpoint;
  end
end;

StoringMaxTrippctMWValue := local_TripPct*(LocalTotal_Available + fPreviousTrippedAmount);
StoringMaxTrippctMWValue := StoringMaxTrippctMWValue - fPreviousTrippedAmount;
if TripMessage <> '' then begin
  TripMessage := TripMessage + ' action triggered. Trip call for '
    + FloatToStr(round(local_TripPct*100)) + '%.';
End;
if StoringMaxTrippctMWValue > 0 then begin // Only trip Gens if the amount is positive
  EVALUATE THIS FUNCTION:
  FindCombinationToTripped(StoringMaxTrippctMWValue, fSimTime, theEventVector,
    TripMessage);
end
else begin
  if TripMessage <> '' then begin
    if fMonitor > 0 then
      Report Event: 'Relay Monitoring Only Will Not Trip. Gen Relay ATR ' + TripMessage
    else
      Report Event: 'Gen Relay ATR ' + TripMessage
    end
  end
end;
end; //if GrossEnable

//-----END OF ALGORITHMS -----//
//----- Individual Unit Over-speed -----//
TripMessage := 'Individual Unit Over-speed';
for IndUnitCt := 1 to 4 do begin
  theGenATR := GenATRRelays[IndUnitCt];
  if (theGenATR.GenObject <> nil) and
     (theGenATR.uXspd >= 1170) and
     (theGenATR.IsInService)
    and (not theGenATR.theUTrip)
  then begin
```



```
if fMonitor > 0 then begin
  Report Event: 'Relay Monitoring Only Will Not Trip. Gen Relay ATR (' +
    TripMessage + ') action triggered. '
  if (not theGenATR.MonitorForTrip) then begin
    theGenATR.theUTrip := true;
    theGenATR.MonitorForTrip := true;
    Local_CurrentTrippedAmount := theGenATR.PGenSystemSetpoint;
    theBus := TTxBusObject(theGenATR.GenObject.SysBusObject);
    Report Event: 'Relay Monitoring Only Will Not Trip. Gen Relay ATR (' +
      TripMessage + ') action trip. ');
    LoadCountIntatBus := theBus.LoadCount;
    LoadCountIntAtFeeder := theBus.LoadsAtEndOfFeeder.count;
    LoadCountTotal := LoadCountIntatBus + LoadCountIntAtFeeder;
    if LoadCountTotal > 0 then begin
      Report Event: 'Relay Monitoring Only Will Not Trip. Loads tripping by 20% after ' +
        'Gen Relay ATR after generator at my bus opened. ');
    end;
    fPreviousTrippedAmount := fPreviousTrippedAmount + Local_CurrentTrippedAmount;
    fAUnitHasTripped := true; // Used in determine if a unit was tripped recently < 2 sec
    fAUnitHasTripped_Time := fSimTime;
  end;
end
else begin
  theGenATR.theUTrip := true;
  Local_CurrentTrippedAmount := theGenATR.PGenSystemSetpoint;
  // Triggered Time
  Report Event: 'Gen ATC Relay (' + TripMessage + ') action triggered. ';
  if (not theGenATR.MonitorForTrip) then begin
    theGenATR.MonitorForTrip := true;
    Initiate Trip Event: theGenATR.GenObject is OPENED
    Initiate Trip Event: Trip 20% of load at the terminal bus of the generator
                        (must account for CMPLDW too)

    // Load Tripping -----//
    fPreviousTrippedAmount := fPreviousTrippedAmount + Local_CurrentTrippedAmount;
    fAUnitHasTripped := true; // Used in determine if a unit was tripped recently < 2 sec
    fAUnitHasTripped_Time := fSimTime;
  end;
end;
end;
end;
//-----End of Individual Unit Over-speed -----//
end;
```



Function FindCombinationToTripped here

Declaration:

```
FindCombinationToTripped(TripMW : single; fSimTime : TPWTimeStamp; theEventVector : TPWVector;  
theTripMessage : string);
```

//List of Local Variables

```
i : integer;  
theComboEntry : TTxComboEntry;  
Local_CurrentTrippedAmount : single;  
currentIndex : integer;  
theGenATRRelay : TTxGenOther_ATRRELAY_Generic_PWGenRelayModel;  
local_Event : TTxEventObject;  
kk : integer;  
theBus : TTxBusObject;  
LoadCountIntatBus, LoadCountIntAtFeeder, LoadCountTotal : integer;  
TheLoad : TTxLoadObject;  
Local_MW : single; // jamie 4/1/20  
SmallestMW : single; // jamie 4/1/20  
SmallestIndex : integer; // jamie 4/1/20  
TrippingAll : boolean; // jamie 4/1/20
```

Actual Code for the function

```
Local_CurrentTrippedAmount := 0; // just initialize this!  
currentIndex := -1;  
for i := 0 to theTripList.count-1 Do Begin  
  theComboEntry := theTripList.Item(i);  
  Local_CurrentTrippedAmount := 0;  
  for kk := 1 to 4 do begin  
    if theComboEntry.UTrip[kk] = 1 then begin  
      theGenATRRelay := GenATRRelays[kk];  
      if theGenATRRelay.IsInService  
      then begin  
        Local_CurrentTrippedAmount := Local_CurrentTrippedAmount +  
          theGenATRRelay.PGenSystemSetpoint;  
        if (theGenATRRelay.MonitorForTrip) then Local_CurrentTrippedAmount := -10000000;  
      end  
      else Local_CurrentTrippedAmount := -10000000; // means that this combination will not be  
        used because it already tripped  
    end;  
  end;  
  if Local_CurrentTrippedAmount >= TripMW then begin  
    currentIndex := i;  
    for kk := 1 to 4 do begin  
      if theComboEntry.UTrip[kk] = 1 then begin  
        if (not GenATRRelays[kk].MonitorForTrip) then GenATRRelays[kk].MonitorForTrip := true;  
      end;  
    end;  
    break;  
  end;  
end;  
if currentIndex > -1 then begin // Found a combination to use  
  theComboEntry := theTripList.Item(currentIndex);  
  Report Event: 'This Combination Was Selected: ' + IntToStr(theComboEntry.UTrip[1]) + ',' +  
    IntToStr(theComboEntry.UTrip[2]) + ',' +  
    IntToStr(theComboEntry.UTrip[3]) + ',' + IntToStr(theComboEntry.UTrip[4])  
    + ' (' + theTripMessage + ')';  
  
  // jamie 4/1/20  
  SmallestIndex := -1;  
  SmallestMW := 0;  
  TrippingAll := False;  
  if fLeaveOneOn then begin  
    TrippingAll := True;  
    for kk := 1 to 4 do begin  
      theGenATRRelay := GenATRRelays[kk];  
      if theGenATRRelay.IsInService then begin
```



```

local_MW := theGenATRRElay.GetTheInitialPGen;
if (SmallestIndex < 0)
  or (local_MW < SmallestMW)
then begin
  SmallestIndex := kk;
  SmallestMW := local_MW;
end;
if theComboEntry.UTrip[kk] <> 1 then TrippingAll := false;
end;
end;
if TrippingAll and (SmallestIndex > 0) then begin
  Report Event: 'Because LeaveOneOn <> 0, Gen Entry ' +
    IntToStr(SmallestIndex) + ' will not be tripped');
end;
end;

for kk := 1 to 4 do begin
  if (theComboEntry.UTrip[kk] = 1)
    and (not (fLeaveOneOn and TrippingAll and (kk = SmallestIndex))) // jamie 4/1/20
  then begin
    fAUnitHasTripped := true; // Used in determine if a unit was tripped recently (less than
      2 seconds ago)
    fAUnitHasTripped_Time := fSimTime;
    if fMonitor > 0 then begin // jamie 1/30/15
      theGenATRRElay := TTxGenOther_ATRRELAY_Generic_PWGenRelayModel(GenATRRelays[kk]);
      if theGenATRRElay.IsInService then begin
        theBus := TTxBusObject(theGenATRRElay.GenObject.SysBusObject);
        Report Event: 'Monitoring Only Will Not Trip.' + 'Gen Relay ' + ModelType + ' (' +
          theTripMessage + '). ');
        Report Event: 'Relay Monitoring Only Will Not Trip.' + 'Gen Relay ' + ModelType + '
          (' + theTripMessage + ') action tripped ' +
          FloatToStr(round(theGenATRRElay.PGenSystemSetpoint*TTxPowerSystem(Owner).SBase)) +
          ' MW. ');
        LoadCountIntatBus := theBus.LoadCount;
        LoadCountIntatFeeder := theBus.LoadsAtEndOfFeeder.count;
        LoadCountTotal := LoadCountIntatBus + LoadCountIntatFeeder;
        if LoadCountTotal > 0 then
          Report Event: 'Relay Monitoring Only Will Not Trip. Loads tripping by 20% after
            Gen Relay ' + ModelType + ' after generator at Bus ' + IntToStr(theBus.num) + '
            opened. ');
        end;
      end;
    end;
  else begin
    If theEventVector <> nil Then Begin // Add the event
      theGenATRRElay := TTxGenOther_ATRRELAY_Generic_PWGenRelayModel(GenATRRelays[kk]);
      if theGenATRRElay.IsInService then begin
        Report Event: 'Relay', ModelType, 'Gen Relay ' + ModelType + ' (' + theTripMessage +
          '). ');
        Initiate Trip Event: theGenATRRElay.GenObject is OPENED
        Initiate Trip Event: Trip 20% of load at the terminal bus of the generator
          (must account for CMPLDW too)
        end; // if theGenATRRElay.IsInService then begin
      end;
    end;
  end;
end;
fPreviousTrippedAmount := fPreviousTrippedAmount + Local_CurrentTrippedAmount;
end;
End;

```