## Condenser Fan Degradation

### Description

def description

return "Motor efficiency degrades when a motor suffers from a bearing or a stator winding fault. This fault causes the motor to draw higher electrical current without changing the fluid flow. Both a bearing fault and a stator winding fault can be modeled by increasing the power consumption of the condenser fan without changing the airflow of the condenser fan. This measure simulates the condenser fan degradation by modifying the Coil:Cooling:DX:SingleSpeed object in EnergyPlus assigned to the heating and cooling system. The fault intensity (F) for this fault is defined as the reduction in motor efficiency as a fraction of the non-faulted motor efficiency."

end

### Modeler Description

def modeler\_description

return "Three user inputs are required and, based on these user inputs, the EIR in the DX cooling coil model is recalculated to reflect the faulted operation as shown in the equation below, where EIRF is the faulted EIR, W ̇\_fan is the fan power, W ̇\_cool is the DX coil power, and F is the fault intensity. EIR\_F/EIR=1+(W ̇\_fan/W ̇\_cool)\*(F/(1-F)). This fault model also requires the ratio of condenser fan power to the power consumption of compressor and condenser fan as a user input parameter."

end

### Measure Type

EnergyPlus Measure

**Taxonomy**

HVAC.Cooling

### Arguments

def arguments(workspace)

args = OpenStudio::Ruleset::OSArgumentVector.new

list = OpenStudio::StringVector.new

list << $all\_coil\_selection

singlespds = workspace.getObjectsByType("Coil:Cooling:DX:SingleSpeed".to\_IddObjectType)

singlespds.each do |singlespd|

list << singlespd.name.to\_s

end

twostages = workspace.getObjectsByType("Coil:Cooling:DX:TwoStageWithHumidityControlMode".to\_IddObjectType)

twostages.each do |twostage|

list << twostage.name.to\_s

end

#make choice arguments for Coil:Cooling:DX:SingleSpeed

coil\_choice = OpenStudio::Ruleset::OSArgument::makeChoiceArgument("coil\_choice", list, true)

coil\_choice.setDisplayName("Enter the name of the faulted Coil:Cooling:DX:SingleSpeed object. If you want to impose the fault on all coils, select #{$all\_coil\_selection}")

coil\_choice.setDefaultValue($all\_coil\_selection)

args << coil\_choice

#choice of schedules for the presence of fault. 0 for no fault and other numbers means fault level

#schedule

sch\_choice = OpenStudio::Ruleset::OSArgument::makeStringArgument("sch\_choice", true)

sch\_choice.setDisplayName("Enter the name of the schedule of the fault level. If you do not have a schedule, leave this blank.")

sch\_choice.setDefaultValue("")

args << sch\_choice #FUTURE: detect empty string later for users who provide no schedule, and delete schedule\_exist

#make a double argument for the fault level

#it should range between 0 and 0.9. 0 means no degradation

#and 0.9 means that percentage drop of COP is 90% and percentage drop of cooling load is also 90%

fault\_lvl = OpenStudio::Ruleset::OSArgument::makeDoubleArgument("fault\_lvl", false)

fault\_lvl.setDisplayName("Fan motor efficiency degradation ratio [-]")

fault\_lvl.setDefaultValue(0.5) #default fouling level to be 50%

args << fault\_lvl

#make a double argument for the fault level

#it should range between 0 and 0.9. 0 means no degradation

#and 0.9 means that percentage drop of COP is 90% and percentage drop of cooling load is also 90%

fan\_power\_ratio = OpenStudio::Ruleset::OSArgument::makeDoubleArgument("fan\_power\_ratio", false)

fan\_power\_ratio.setDisplayName("Ratio of condenser fan motor power consumption to combined power consumption of condenser fan and compressor at rated condition.")

fan\_power\_ratio.setDefaultValue(0.091747081) #defaulted calcualted to be 0.0917

args << fan\_power\_ratio

return args

end

### Initial Condition

#Select fan object that is being faulted.

runner.registerInitialCondition("Imposing performance degradation on "+coil\_choice\_all+".")

### Final Condition

#Impose efficiency degradation on the fan object.

runner.registerFinalCondition("Imposed performance degradation on "+coil\_choice\_all+".")

### Not Applicable

#When the fault level is not defined,

runner.registerAsNotApplicable("CondenserFanDegradation is not running for "+coil\_choice\_all+". Skipping......")

### Warning

n/a

### Error

#When fault intensity schedule is not defined,

runner.registerError("User-defined schedule "+sch\_choice+" does not exist. Exiting......")

#When fault intensity schedule is defined but the range is outside the limit (0-1),

runner.registerError("User-defined schedule "+sch\_choice+" has a ScheduleTypeLimits outside the range 0 to 1.0. Exiting......")

#When fault intensity constant value is defined but the range is outside the limit (0-1),

runner.registerError("Fault level #{degrd\_lvl} for "+coil\_choice\_all+" is oustide the range from 0 to 0.99. Exiting......")

#When Condenser Type option of DX unit is not defined as AirCooled

runner.registerError(coil\_choice+" is not air cooled. Impossible to impose condenser fan motor efficiency degradation. Exiting......")

#When EIR performance curve is not defined,

runner.registerError("No Temperature Adjustment Curve for "+coil\_choice+" EIR. Exiting......")

#When the DX unit cannot be found,

runner.registerError("Measure CondenserFanDegradation cannot find "+coil\_choice\_all+". Exiting......")

### Information

* Works with,
  + Coil:Cooling:DX:SingleSpeed
  + Coil:Cooling:DX:TwoStageWithHumidityControlMode.
* Future refinement items are,
  + Capability to work with other DX models.
  + Capability of generic autosizing to hardsizing.

### Code Outline

* Define arguments (DX unit where fault occurs, fault level in constant value or scheduled values, fan power ratio in constant value).
* Check scheduled fault level values (within 0-1) if exists.
* Check constant fault level value (within 0-1).
* Find the DX unit where fault occurs.
* Create string object in idf (with EMS) for fault implementation.
  + Create fractional schedule object for fault level implementation (use fault level values either from the constant or scheduled input arguments).
  + Create schedule for both non-faulted and faulted period.
  + Append EMS code for EIR adjustment calculation under faulted operation.
  + Append EMS code for defining EMS sensor object.
  + Append EMS code for defining EMS output object.

### Tests

* Test model with and without schedule of fault presence
* Test invalid user argument values to make sure measure fails gracefully