## Air Handling Unit Fan Motor Degradation

### Description

def description

return "Fan motor degradation occurs due to bearing and stator winding faults, leading to a decrease in motor efficiency and an increase in overall fan power consumption. This measure simulates the air handling unit fan motor degradation by modifying either the Fan:ConstantVolume, Fan:VariableVolume, or the Fan:OnOff objects in EnergyPlus assigned to the ventilation system. The fault intensity (F) for this fault is defined as the ratio of fan motor efficiency degradation."

end

### Modeler Description

def workspaceer\_description

return "Two user inputs are required and, based on these user inputs, the fan efficiency is recalculated to reflect the faulted operation as shown below, where η\_(fan,tot,F) is the degraded total efficiency under faulted condition, η\_(fan,tot) is the total efficiency under normal condition, and F is the fault intensity. η\_(fan,tot,F) = η\_(fan,tot)∙(1-F)"

end

### Measure Type

EnergyPlus Measure

**Taxonomy**

HVAC.Ventilation

### Arguments

def arguments(workspace)

args = OpenStudio::Ruleset::OSArgumentVector.new

list = OpenStudio::StringVector.new

list << $allchoices

cvs = workspace.getObjectsByType("Fan:ConstantVolume".to\_IddObjectType)

cvs.each do |cv|

list << cv.name.to\_s

end

ofs = workspace.getObjectsByType("Fan:OnOff".to\_IddObjectType)

ofs.each do |of|

list << of.name.to\_s

end

vvs = workspace.getObjectsByType("Fan:VariableVolume".to\_IddObjectType)

vvs.each do |vv|

list << vv.name.to\_s

end

# make choice arguments for fan

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

fan\_choice.setDisplayName("Enter the name of the faulted Fan:ConstantVolume, Fan:OnOff object or Fan:VariableVolume. If you want to impose the fault on all fan objects in the building, enter #{$allchoices}")

fan\_choice.setDefaultValue($allchoices)

args << fan\_choice

# make a double argument for the fault level

# it should range between 0 and 1. 0 means no degradation

eff\_degrad\_fac = OpenStudio::Ruleset::OSArgument.makeDoubleArgument('eff\_degrad\_fac', false)

eff\_degrad\_fac.setDisplayName('Degradation factor of the total efficiency of the fan during the simulation period. If the fan is not faulted, set it to zero.')

eff\_degrad\_fac.setDefaultValue(0.15) # default fouling level to be 15%

args << eff\_degrad\_fac

# choice of schedules for the presence of fault. 0 for no fault and 1 means total degradation

sch\_choice = OpenStudio::Ruleset::OSArgument.makeStringArgument('sch\_choice', false)

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

return args

end

### Initial Condition

#Select fan object that is being faulted.

runner.registerInitialCondition("Imposing airflow restriction on #{fan\_choice}.")

### Final Condition

#Impose efficiency degradation on the fan object.

runner.registerFinalCondition("Imposed efficiency degradation level at #{eff\_degrad\_fac} on #{fan\_choice}.")

### Not Applicable

n/a

### 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("Fan Efficiency Degradation Level #{eff\_degrad\_fac} for #{fan\_choice} is outside the range 0 to 1.0. Exiting......")

#When selected fan cannot be found in the model,

runner.registerError("Measure FanMotorDegradation cannot find #{fan\_choice}. Skipping......")

### Information

n/a

### Code Outline

* Define arguments (air handling unit where fault occurs, fault level in constant value or scheduled values).
* Check scheduled fault level values (within 0-1) if exists.
* Check constant fault level value (within 0-1).
* Create fractional schedule object for fault level implementation (use fault level values either from the constant or scheduled input arguments).
* Find the fan object(s) assigned to the air handling unit that was selected as argument.
* Store original efficiency values from the fan object(s).
* Write EMS program to impose degraded efficiency for each fan object.
* Define sensor object (storing efficiency degradation values in fractional schedule).
* Define program object (calculate fan efficiency after degradation).
* Define ProgramCallingManager object (define EMS calling point).
* Define Actuator object (apply degraded efficiency to fan object(s)).
* Define EMS output object

### Tests

* Test model with Fan:ConstantVolume
* Test model with Fan:OnOff
* Test model with Fan:VariableVolume
* Test invalid user argument values to make sure measure fails gracefully
* Test fault intensity with constant value
* Test fault intensity with scheduled values