## Duct Fouling

### Description

def description

return " Ducts are fouled by dust that accumulates in the filter and/or fins of heat exchangers in the indoor air ducts. The accumulation increases the flow resistance of the air duct and changes the airflow and pressure drop across the duct in accordance with the controls of the fan rotational speed. This fault is categorized as a fault that occur in the ventilation system during the operation stage. This fault measure is based on an empirical model and simulates duct fouling by modifying either Fan:ConstantVolume, Fan:VariableVolume, Fan:OnOff, or Fan:VariableVolume objects in EnergyPlus assigned to the air system. F is the fault intensity defined as the reduction in evaporator coil airflow at full load condition as a ratio of the design airflow rate with the application range of 0 to 0.5 (50% reduction)."

end

### Modeler Description

def modeler\_description

return "Two additional user inputs are required. Based on these user inputs, the maximum supply airflow rate parameter defined in fan objects is replaced based on equation, mdot\_(a,max,F) = mdot\_(a,max)∙(1-F), where mdot\_(a,max,F) is the maximum airflow rate of the faulted condition, mdot\_(a,max) is the maximum airflow rate under normal conditions, and F is the fault intensity defined as the reduction in evaporator coil airflow at full load condition as a ratio of the design airflow rate. There is a pressure rise (r\_pd) parameter that is also required in fan objects in order to properly reflect evaporator fouling. Equation, F = 1-√((1+r\_pd-c\_F)/(1-c\_F )) shows the relation between F and r\_pd that is used to calculate the pressure rise based on the fault intensity level. cF is the coefficient that is determined based on the training data set."

end

### Measure Type

OpenStudio Measure

**Taxonomy**

HVAC.Ventilation

### Arguments

def arguments(model)

args = OpenStudio::Ruleset::OSArgumentVector.new

# find all the airloop objects

airloophvacs = model.getAirLoopHVACs

chs = OpenStudio::StringVector.new

chs << $allahuchoice

airloophvacs.each do |airloophvac|

chs << airloophvac.name.to\_s

end

equip\_name = OpenStudio::Ruleset::OSArgument.makeChoiceArgument('equip\_name', chs, true) # use the names for choices of equipment

equip\_name.setDisplayName('Choice of AirLoopHVAC objects. If you want to impose it on all AHUs, choose \* ALL AHUs \*')

equip\_name.setDefaultValue($allahuchoice)

args << equip\_name

# ask user for a fault level in terms of the percentage of mass flow rate reduction

evap\_flow\_reduction = OpenStudio::Ruleset::OSArgument.makeDoubleArgument('evap\_flow\_reduction', true)

evap\_flow\_reduction.setDefaultValue(0.1)

evap\_flow\_reduction.setDisplayName('Decrease of air mass flow rate ratio when the fans are running at their maximum speed (0-1). (-)')

args << evap\_flow\_reduction

#make double arguments to obtain coefficients

args = enter\_coefficients(args, 1, 'fanCurve', [1.4048])

return args

end

### Initial Condition

#Select AirLoopHVAC object that is being faulted.

runner.registerInitialCondition('Fouling are being applied on all AHUs......')

runner.registerInitialCondition(“Fouling is being applied to the #{equip\_name}......")

### Final Condition

#Impose performance degradation due to evaporator fouling.

runner.registerFinalCondition('Fouling are applied on all AHUs......')

runner.registerFinalCondition("Fouling is applied to the #{equip\_name}......")

### Not Applicable

#When the fault level is defined as zero,

runner.registerAsNotApplicable("Fouling level is zero. Skipping the Measure #{name}")

### Warning

n/a

### Error

#When reduction of evaporator flow is negative,

runner.registerError("User defined fouling level in Measure #{name} is negative. Exiting......")

#When reduction of evaporator flow is greater than 1,

runner.registerError("The resultant mass flow rate in Measure #{name} is negative. Exiting......")

#When minimum fan power for variable speed fan cannot be calculated,

runner.registerError("Cannot find the airflow corresponding to the minimum power consumption. Exiting......")

#When equation for solving “ ” cannot be solved,

runner.registerError("Dekker method fails with x\_high at #{x\_high}, x\_low at #{x\_low}, y\_new at #{y\_new} and mulp at #{mulp}. Exiting......")

### Information

* Works with,
  + Fan:ConstantVolume
  + Fan:VariableVolume
  + Fan:OnOff

### Code Outline

* Define arguments (AHU where fault occurs and flow reduction due to fouling).
* Check user defined flow reduction value due to fouling is within 0-1.
* Find the AHU and apply fault based on user inputs.
  + Find Fan:ConstantVolume object in the AHU and apply fault.
    - If the fan flow rate is hard sized, then skip.
    - Else, change the fan configuration based on user defined fault intensity.
      * Pressure rise.
      * Maximum flow rate.
  + Find Fan:VariableVolume object in the AHU and apply fault.
    - If the fan flow rate is hard sized, then skip.
    - Else, change the fan configuration based on user defined fault intensity.
      * Pressure rise.
      * Maximum flow rate.
      * Fan power minimum flow fraction.
      * Fan power minimum air flow rate.
  + Find Fan:OnOff object in the AHU and apply fault.
    - If the fan flow rate is hard sized, then skip.
    - Else, change the fan configuration based on user defined fault intensity.
      * Pressure rise.
      * Maximum flow rate.

### Tests

* Test invalid user argument values to make sure measure fails gracefully.
* Test three fan objects (Fan:ConstantVolume, Fan:VariableVolume, Fan:OnOff).