R Notebook

This is an R Markdown Notebook. When you execute code within the notebook, the results appear beneath the code.

Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Ctrl+Shift+Enter*.

Setup

Knowns

Given we have a bunch of known values:

- $\dot{q}_{IT}$ = steady-state IT load
- $\dot{q}^{max}$ = nominal capacity of the cooling system
- $\dot{C}_h$ = capacity rate of the hot fluid (air)
- ΔT_{ch} = nominal temperature difference across the chiller
- T_s^a = nominal supply temperature of the air from the CRAH
- T_{stor} = nominal supply temperature of the water from the chiller

Assumptions

- $\dot{C}_h^{ref} = \dot{C}_h$
- $\dot{C}_{ch} = \frac{\dot{q}_{IT}}{\Delta T_{ch}}$
- $\dot{C}_{cc} = \dot{C}_{ch} \frac{\dot{q}_{IT}}{\dot{q}^{max}}$

HEX model

$$\epsilon(\dot{C}_c, \dot{C}_h) = 1 - \frac{1 - \dot{C}_{\min}/\dot{C}_{\max}}{\left(\frac{1-\epsilon_{ref}(\dot{C}_{\min}^{ref}/\dot{C}_{\max}^{ref})}{1-\epsilon_{ref}}\right)^{\alpha(\dot{C}_c, \dot{C}_h)} - \dot{C}_{\min}/\dot{C}_{\max}}$$

$$\alpha(\dot{C}_c, \dot{C}_h) = \frac{2\left(\frac{\dot{C}_c}{\dot{C}_c^{ref}}\right)^{\frac{4}{5}} \left(\frac{\dot{C}_h}{\dot{C}_h^{ref}}\right)^{\frac{4}{5}} \dot{C}_{\min}^{ref}(1 - \dot{C}_{\min}/\dot{C}_{\max})}{\left(\left(\frac{\dot{C}_c}{\dot{C}_c^{ref}}\right)^{\frac{4}{5}} + \left(\frac{\dot{C}_h}{\dot{C}_h^{ref}}\right)^{\frac{4}{5}}\right) \dot{C}_{\min}(1 - \dot{C}_{\min}^{ref}/\dot{C}_{\max}^{ref})}$$

Our Problem

We want to find values for $\dot{C}_c^{ref}$ and ϵ_{ref}, so that the HEX removes the "right" amount of heat in two specific conditions (when $\dot{C}_c = \dot{C}_{cc}$, normal SS operating conditions, and $\dot{C}_c = \dot{C}_{ch}$, maximum capacity of the cooling system conditions) and also provides physically sensible effectiveness *under normal operating conditions*.

Equations

We want to find values for $\dot{C}_c^{ref}$ and ϵ_{ref} by solving the following two equations.

Heat removed when the maximum water flow rate is sent through the coil

$$\epsilon(\dot{C}_{ch}, \dot{C}_h) = \frac{\dot{q}^{max}}{\min(\dot{C}_{ch}, \dot{C}_h)(T_s^a - \dot{q}^{max}/\dot{C}_h + T_{stor})}$$

Heat removed when the steady-state amount of water is sent through the coil

$$\epsilon(\dot{C}_{cc}, \dot{C}_h) = \frac{\dot{q}^{IT}}{\min(\dot{C}_{ch}, \dot{C}_h)(T_s^a - \dot{q}^{IT}/\dot{C}_h + T_{stor})}$$

Additional Considerations

Just because we can solve the previous two equations for the unknowns $\dot{C}_c^{ref}$ and ϵ_{ref}, it doesn't mean that these values can provide physically sensible values. For example, we can't always guarantee that $\dot{C}_c^{ref} > 0$ and $\epsilon_{ref} \in [0, 1]$ (or even that $\epsilon(\dot{C}_{ch}, \dot{C}_h), \epsilon(\dot{C}_{cc}, \dot{C}_h) \in [0, 1]$).

We want to find the minimal ΔT_{ch} that will allow for physically sensible values:

- Ensure that $\epsilon(\dot{C}_c, \dot{C}_h)$ is between 0 and 1 for all reasonable values of $\dot{C}_c$ and $\dot{C}_h$
  - $\dot{C}_c$ should be between 0 and $\dot{C}_{ch}$
  - $\dot{C}_h$ normally takes one of two values $\dot{C}_h^{ref}$ and $\dot{C}_h^{ref}/AR$ where AR is the air ratio.

In many cases, the nominal ΔT_{ch} may be good enough to ensure physically sensible values. However, in scenarios where T_s^a is not that much bigger than T_{stor} or $\dot{C}_{cc} << \dot{C}_h$, we may need to reduce the ΔT_{ch}.

Sample Code for Table Search

Please not that this is code that currently finds an approximate pairing of $\dot{C}_c^{ref}$ and ϵ_{ref}. It doesn't not try to achieve physically sensible values by lowering the ΔT_{ch}

```
q_it <- 30000
q_max <- 75000
C_h_ref <- 4000
deltaT_ch <- 2
c_p_w <- 4217
c_p_a <- 1005
C_ch <- q_it / deltaT_ch
C_cc_ss <- C_ch * q_it/q_max
T_s_a <- 16
T_stor <- 15
T_r_a <- T_s_a + q_it/C_h_ref

print(C_cc_ss)
```

```
## [1] 6000
```

```
print(1/0.80*C_h_ref*(T_r_a-T_s_a)/(T_r_a-T_stor))
```

```
## [1] 4411.765
```

```
print((T_r_a-T_s_a)/(T_r_a-T_stor))
```

```
## [1] 0.8823529
```

Problem Setup

```
alpha <- function(C_c,C_c_ref,e_ref){
  C_min = min(C_c, C_h_ref)
  C_max = max(C_c, C_h_ref)
  C_min_ref = min(C_c_ref, C_h_ref)
  C_max_ref = max(C_c_ref, C_h_ref)
  alpha = 2*(C_c/C_c_ref)^(4/5)*(C_h_ref/C_h_ref)^(4/5)*C_min_ref*(1-C_min/C_max)/(((C_c/C_c_ref)^(4/5)
}

epsilon <- function(C_c, C_c_ref, e_ref){
  C_min = min(C_c, C_h_ref)
  C_max = max(C_c, C_h_ref)
  C_min_ref = min(C_c_ref, C_h_ref)
  C_max_ref = max(C_c_ref, C_h_ref)
  a = alpha(C_c,C_c_ref,e_ref)
  epsilon = 1-(1-C_min/C_max)/(((1-e_ref*(C_min_ref/C_max_ref))/(1-e_ref))^a-C_min/C_max)
}
```

Compute Errors

```
num_of_values_C <- 50
num_of_values_e <- 50

C_c_ref_values <- seq(0.001,6*C_h_ref,length.out = num_of_values_C)
#C_c_ref_values <- C_h_ref+100
e_ref_values <- seq(0.001,0.999,length.out = num_of_values_e)
e_1 = q_max/(min(C_ch,C_h_ref)*(T_s_a+q_max/C_h_ref-T_stor))
e_2 = q_it/(min(C_cc_ss,C_h_ref)*(T_s_a+q_it/C_h_ref-T_stor))
C_array <- NULL
e_array <- NULL
error_array <- NULL

for(C_ref in C_c_ref_values){
  for(e_ref in e_ref_values){
    error = 0
    e_1_val= epsilon(C_ch,C_ref,e_ref)
    e_2_val = epsilon(C_cc_ss,C_ref,e_ref)
    error = error + (e_1-e_1_val)^2
    error = error + (e_2-e_2_val)^2
    C_array = c(C_array,C_ref)
    e_array = c(e_array,e_ref)
    #if (!is.nan(error)){
      error_array = c(error_array,sqrt(error))
    #} else {
    #  error_array = c(error_array,999)
    #}
  }
}
min_index = which.min(error_array)
print(e_1)
```

```
## [1] 0.9493671
```

```r
print(e_2)
```

```
## [1] 0.8823529
```

```r
print(C_array[min_index])
```

```
## [1] 979.5928
```

```r
print(e_array[min_index])
```

```
## [1] 0.999
```

```r
print(error_array[min_index])
```

```
## [1] 0.01588212
```

Plot results

```r
# require(plotly)
# df<-data.frame(C_array,e_array,error_array)
# p <- plot_ly(z = as.matrix(df), type = "surface")%>%
#   layout(
#     xaxis = list(range = c(0, 1)))
#print(p)
```

Contour Plot

```r
require(ggplot2)
```

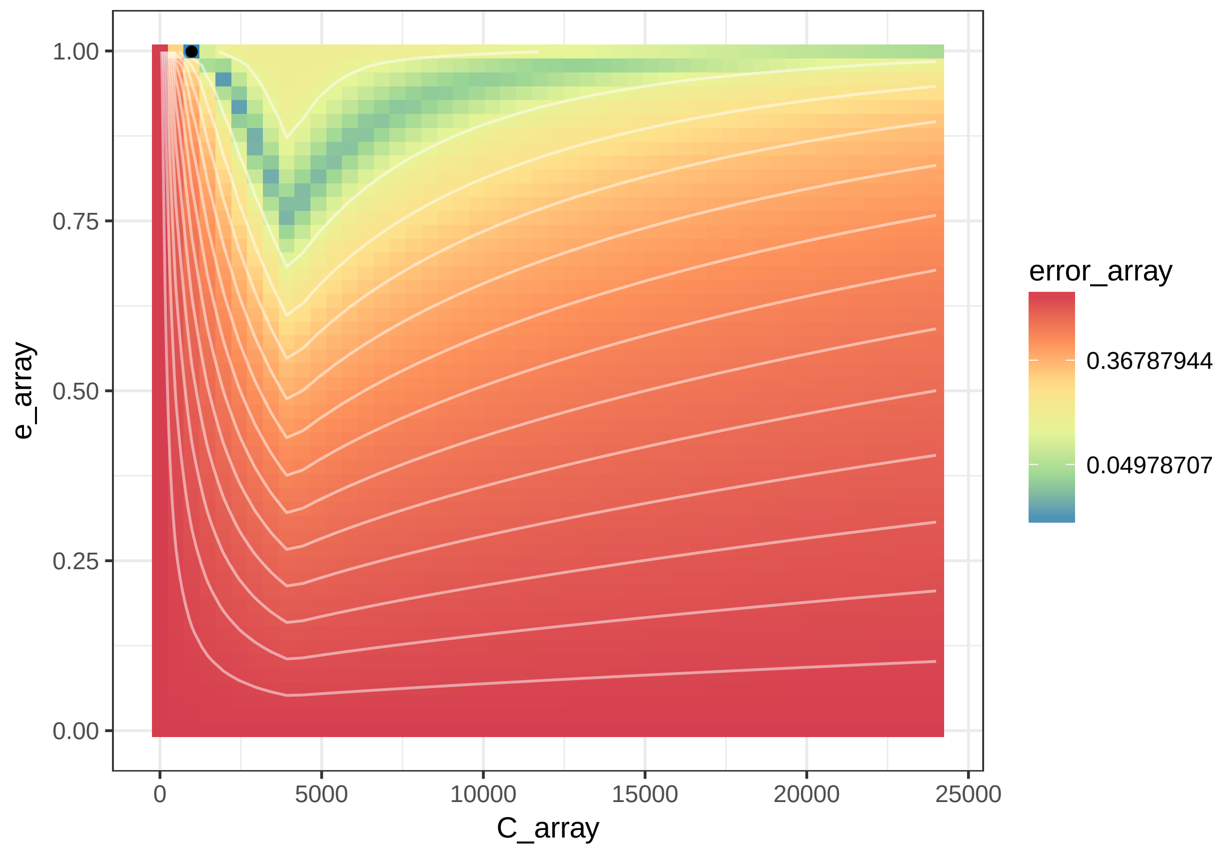
```
## Loading required package: ggplot2

##
## Attaching package: 'ggplot2'

## The following object is masked _by_ '.GlobalEnv':
##
##     alpha
```

```r
df<-data.frame(C_array,e_array,error_array)
g <- ggplot(df,aes(x=C_array,y=e_array,z=error_array))+geom_contour() + geom_raster(aes(fill = error_ar
#print(g)
```

```r
df<-data.frame(C_array,e_array,error_array)
bestdf<-data.frame(C_array[min_index],e_array[min_index],error_array[min_index])
g <- ggplot(df,aes(x=C_array,y=e_array,z=error_array, fill=error_array))+ geom_tile() +
  geom_contour(color = "white", alpha = 0.5) +
  scale_fill_distiller(palette="Spectral", na.value="white", trans = "log") +
  theme_bw() + geom_point(data=bestdf,aes(x=C_array[min_index],y=e_array[min_index],z=error_array[min_i
```

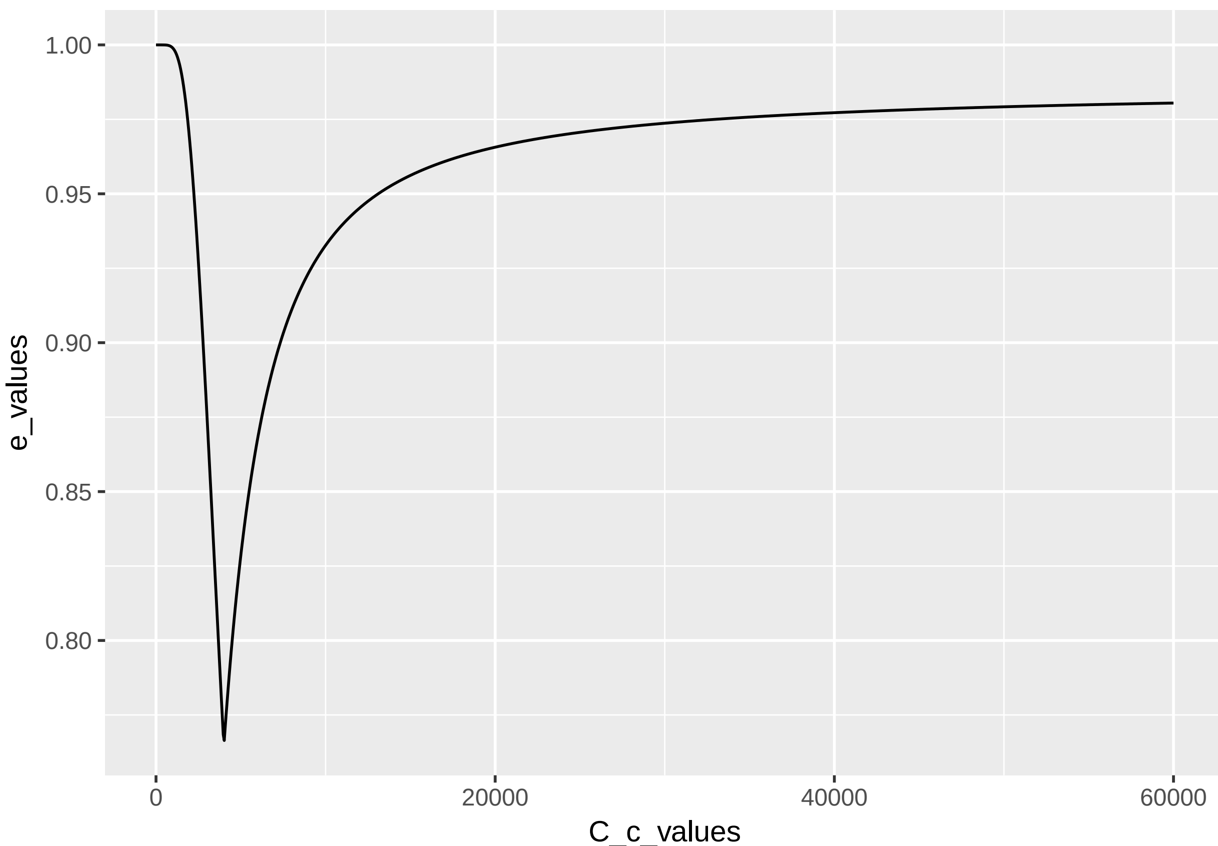
```
## Warning: Ignoring unknown aesthetics: z
```

```r
print(g)
```



Finding a flowrate to remove a certain amount of energy from the air

```
C_c_ref = C_array[min_index]
e_ref = e_array[min_index]
num_of_values_C = 1000

C_c_values <- seq(0.001,4*C_ch,length.out = num_of_values_C)
e_values <- NULL

for(C in C_c_values){
  e = epsilon(C,C_c_ref,e_ref)
  e_values <- c(e_values,e)
}

ef <- data.frame(C_c_values,e_values)
ggplot(ef,aes(C_c_values,e_values))+geom_line()
```



A computational problem appears to be that for any given δT we may see two possible solutions, subdivided into two pieces ($\dot{C}_c < \dot{C}_h$ and $\dot{C}_c > \dot{C}_h$). I don't think this will be too much of a problem if we keep the search to the local area around the old water flowrate (and the timestep is small).

Additionally, when $\dot{C}_c > \dot{C}_h$,

$$\epsilon = \frac{\dot{C}_c \left(T_{co} - T_{stor}\right)}{\dot{C}_{min} \left(T_r^a - T_{stor}\right)}$$

doesn't give us any information on correct values of $\dot{C}_c$ and T_{co}