

Aluno: Raphael Henrique Braga Leivas

Matrícula: 2020028101

Professor Responsável: Márcio Ziviani

Código fonte LaTeX desse arquivo pode ser visto em meu GitHub pessoal:

1 Problema

Precisamos criar um algoritmo para determinar a temperatura de equilíbrio da superfície da parede esquerda T_e de uma placa vertical. Temos as seguintes informações dadas:

- Condutividade térmica: $k = 2,5 \text{ W/m}^2\text{K}$
- Comprimento: $W = 5,0 \text{ m}$
- Altura: $H = 2,0 \text{ m}$
- Espessura: $L = 0,25 \text{ m}$
- Velocidade do ar ambiente na face da parede: $u_\infty = 3 \text{ m/s}$
- Temperatura do ar ambiente na face da parede: $T_\infty = 300 \text{ K}$
- Fluxo de calor prescrito sobre a parede esquerda: $q_p'' = 750 \text{ W/m}^2$
- Temperatura da parede direita: $T_d = 350 \text{ K}$

2 Solução

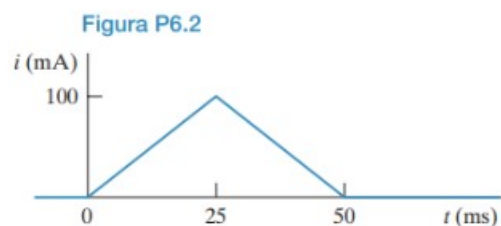
De posse dessas informações, fazemos um desenho esquemático do problema, exibido na Figura 2.1.

Figura 2.1: Diagrama do problema posto, com superfície de controle destacada.

6.2 O pulso triangular de corrente mostrado na Figura P6.2 é aplicado a um indutor de 500 mH.

Pspice
Multisim

- Escreva as expressões que descrevem $i(t)$ nos quatro intervalos $t < 0$, $0 \leq t \leq 25 \text{ ms}$, $25 \text{ ms} \leq t \leq 50 \text{ ms}$ e $t > 50 \text{ ms}$.
- Deduza as expressões para a tensão, potência e energia do indutor. Use a convenção passiva.



Fonte: elaboração própria.

Como mostra a Figura 2.1, temos uma superfície de controle na parede esquerda. Aplicando a primeira lei da termodinâmica sobre essa superfície, temos

$$\dot{E}_{in} + \dot{E}_g - \dot{E}_{out} = \dot{E}_{st} \quad (2.1)$$

Como temos uma superfície de controle, não há energia armazenada e energia gerada. Além disso, no modelo da Figura 2.1, repare que todos os fluxos de calor estão entrando na superfície, de tal maneira que $\dot{E}_{out} = 0$. Assim, (2.1) se reduz a

$$\dot{E}_{in} = 0 \quad (2.2)$$

Expandindo o termo \dot{E}_{in} ,

$$q_p'' + q_c'' + q_k'' = 0$$

$$q_p'' + h_c(T_\infty - T_e) + \frac{k}{L}(T_d - T_e) = 0$$

Isolando o coeficiente convectivo h_c , temos

$$h_c(T_\infty - T_e) = -q_p'' - \frac{k}{L}(T_d - T_e)$$

$$h_c = -\frac{q_p'' + \frac{k}{L}(T_d - T_e)}{T_\infty - T_e} \quad (2.3)$$

Como a velocidade do ar ambiente na face da parede é $u_\infty = 3 \text{ m/s}$, temos que isso equivale a 10,8 km/h. Para ter uma noção do quão rápido essa velocidade é, usamos a Tabela 2.1.

Tabela 2.1: Relação entre a velocidade do vento e seu efeito.

Nº de Beaufort	Descrição	Velocidade do vento (km/h)
0	Calmo	< 1.6
1	Ar leve	1.6 a 4.8
2	Brisa leve	6.4 a 11.2

Fonte: Traduzido de National Oceanic and Atmospheric Administration (NOAA). Disponível em: <https://www.weather.gov/pqr/wind>.

Assim, segundo a Tabela 2.1, temos que $u_\infty = 10,8 \text{ km/h}$ equivale a apenas uma brisa leve, e portanto podemos considerar a convecção como natural. Nota: tentamos considerar a convecção como forçada, mas encontramos erros, conforme elaborado na seção 3.

Uma vez feita a análise inicial do problema, agora criamos um código em linguagem R para calcular iterativamente a temperatura da face esquerda iterativamente até que ela conergir para para temperatura de equilíbrio. O primeiro passo é adicionar os dados do problema no código

```
# dados do problema
k <- 2.5
W <- 5
H <- 2
L <- 0.25
u_inf <- 3
T_inf <- 300
q_pres <- 750
T_d <- 350
```

Em seguida, definimos a tabela com as propriedades termofísicas do ar conforme o livro do Incropera.

```
Properties_Table = matrix(
  c(
    100, 3.5562, 71.1 * 10^-7, 2.00 * 10^-6, 9.34 * 10^-3, 0.786,
    150, 2.3364, 103.4 * 10^-7, 4.426 * 10^-6, 13.8 * 10^-3, 0.758,
```

```

200, 1.7458, 132.5 * 10^-7, 7.590 * 10^-6, 18.1 * 10^-3, 0.737,
250, 1.3947, 159.6 * 10^-7, 11.44 * 10^-6, 22.3 * 10^-3, 0.720,
300, 1.1614, 184.6 * 10^-7, 15.89 * 10^-6, 26.3 * 10^-3, 0.707,
350, 0.9950, 208.2 * 10^-7, 20.92 * 10^-6, 30.0 * 10^-3, 0.700,
400, 0.8711, 230.1 * 10^-7, 26.41 * 10^-6, 33.8 * 10^-3, 0.690,
450, 0.7740, 250.7 * 10^-7, 32.39 * 10^-6, 37.3 * 10^-3, 0.686,
500, 0.6964, 270.1 * 10^-7, 38.79 * 10^-6, 40.7 * 10^-3, 0.684,
550, 0.6329, 288.4 * 10^-7, 45.57 * 10^-6, 43.9 * 10^-3, 0.683,
600, 0.5804, 305.8 * 10^-7, 52.69 * 10^-6, 46.9 * 10^-3, 0.685
),
ncol = 6,
byrow = TRUE
)

colnames(Properties_Table) <- c('Tf', 'rho', 'mu', 'v', 'kf', 'Pr')
rownames(Properties_Table) <- seq(1, nrow(Properties_Table), 1)
Properties_Table <- as.table(Properties_Table)

```

Agora definimos uma variável `max_iterations` que limita o número máximo de iterações para que T_e converja, a fim de evitar loops infinitos em tempo de compilação. Além disso, definimos o valor inicial ("chute") para T_e , e o adicionamos em uma lista que salva os valores calculados de T_e em cada iteração. Consideramos o valor inicial de T_e como a temperatura ambiente.

```

max_iterations <- 100
T_e_calculated <- 298 # chute inicial: Te = 298K (ambiente)
T_e_list <- c(T_e_calculated)

```

Agora definimos um loop `for`, em que cada iteração se calcula um valor de T_e até que ele converja.

```

for (i in 1:max_iterations) {

}

```

Dentro de cada iteração do loop, inicializamos uma variável T_f , que é a temperatura do fluido em que olharemos na tabela, definida por

$$T_f = \frac{T_e + T_\infty}{2} \quad (2.4)$$

No código, temos

```

Tf <- (T_e_calculated + T_inf) / 2

```

O próximo passo é percorrer a tabela das propriedades termofísicas, identificando as temperaturas T_{fmin} e T_{fmax} entre as quais a temperatura T_f se encontra, isto é, a faixa $T_{fmin} \leq T_f \leq T_{fmax}$.

```

# iniciais
Tf_min <- Properties_Table[1, 'Tf']
Tf_min_index <- 1
Tf_max <- Properties_Table[nrow(Properties_Table), 'Tf']
Tf_max_index <- nrow(Properties_Table)

# procura na tabela alguém com esse valor de Tf
for (j in 1:nrow(Properties_Table)) {
  if (Properties_Table[j, 'Tf'] <= Tf) {
    Tf_min <- Properties_Table[j, 'Tf']
    Tf_min_index <- j
  }
}

```

```

    if (Properties_Table[j, 'Tf'] > Tf) {
      Tf_max <- Properties_Table[j, 'Tf']
      Tf_max_index <- j
      break
    }
  }
}

```

Isso deve ser feito pois T_f pode não coincidir exatamente com os valores tabelados. Assim, usamos a variável `interpolation_ratio` para realizar a interpolação dos valores das propriedades, dependendo do quão próximo T_f está de $T_{f_{min}}$ e $T_{f_{max}}$.

```

# pega a razao que diz o quao proximo T_f esta de min ou max
interpolation_ratio <- (Tf - Tf_min) / (Tf_max - Tf_min)

```

Com a razão de interpolação, calculas as propriedades termofísicas necessárias para os cálculos através de interpolação com os dados da tabela.

```

rho_min <- Properties_Table[Tf_min_index, 'rho']
rho_max <- Properties_Table[Tf_max_index, 'rho']
rho <- rho_min + (rho_max - rho_min) * interpolation_ratio

mu_min <- Properties_Table[Tf_min_index, 'mu']
mu_max <- Properties_Table[Tf_max_index, 'mu']
mu <- mu_min + (mu_max - mu_min) * interpolation_ratio

v_min <- Properties_Table[Tf_min_index, 'v']
v_max <- Properties_Table[Tf_max_index, 'v']
v <- v_min + (v_max - v_min) * interpolation_ratio

kf_min <- Properties_Table[Tf_min_index, 'kf']
kf_max <- Properties_Table[Tf_max_index, 'kf']
kf <- kf_min + (kf_max - kf_min) * interpolation_ratio

Pr_min <- Properties_Table[Tf_min_index, 'Pr']
Pr_max <- Properties_Table[Tf_max_index, 'Pr']
Pr <- Pr_min + (Pr_max - Pr_min) * interpolation_ratio

```

Com as propriedades termofísicas calculadas, agora calculamos os coeficientes adimensionais

```

Re_critical <- 50000
Re <- u_inf * L * rho / mu
Nu <- 0.0
Gr <- 9.81 * (1/Tf) * (abs(T_e_calculated - T_inf) * L^3) / (v^2)
Ra <- Gr * Pr

Nu <- 0.825 + (0.387 * Ra^(1/6)) / ((1 + (0.492/Pr)^(9/16))^(8/27))^2

```

Note que, para o cálculo do Número de Nusselt, usamos a expressão (2.5), que é a mesma para escoamentos laminar e turbulento em uma placa plana vertical.

$$Nu = \left\{ 0,825 + \frac{0,387Ra^{1/6}}{\left[1 + (0,492/Pr)^{9/16}\right]^{8/27}} \right\}^2 \quad (2.5)$$

Encontramos o coeficiente convectivo h_c através de

$$h_c = \frac{Nu \cdot K_f}{L} \quad (2.6)$$

No código,

```
hc <- Nu * kf / L
```

Com h_c calculado, encontramos a temperatura da face esquerda da parede diretamente através da expressão (2.3). Isolando T_e nessa equação, temos

$$h_c T_\infty - h_c T_e = -q_p'' - \frac{k}{L} T_d + \frac{k}{L} T_e$$

$$-T_e \left(h_c + \frac{k}{L} \right) = -h_c T_\infty - q_p'' - \frac{k}{L} T_d$$

$$T_e = \frac{h_c T_\infty + q_p'' + \frac{k}{L} T_d}{h_c + \frac{k}{L}} \quad (2.7)$$

No código, calculamos T_e através de (2.7) e o inserimos na lista de T_e calculados

```
T_e_calculated <- ((k/L) * T_d + q_pres + T_inf * hc) / (hc + (k/L))
T_e_list <- append(T_e_list, T_e_calculated)
```

Ao final da iteração, verificamos se o valor calculado está dentro da tolerância de 0,1% definida. Caso esteja dentro dessa tolerância, já está convergindo e deve sair do loop com o break. Caso contrário, continua para a próxima iteração do loop, com next.

```
if (abs(T_e_list[i + 1] - T_e_list[i]) < tolerance * T_e_list[i]) {
  break
} else {
  next
}
```

Finalmente, ao final das iterações do loop, plotamos os valores de T_e armazenados na lista, para que possamos visualizar a convergência.

```
plot(
  T_e_list,
  main = "T_e (K) x Iteracao",
  xlab = "Iteracao",
  ylab = "T_e (K)",
  col = "black",
  lwd = 3
)

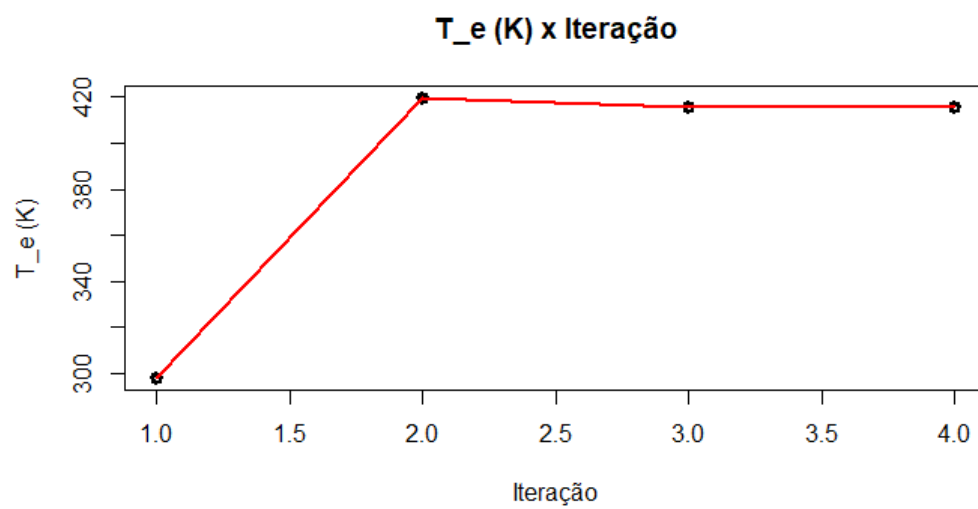
lines(T_e_list, col = "red", lwd = 2, lty = 1)
```

O gráfico plotado pelo software está exibido na Figura 2.2. A temperatura de equilíbrio para a superfície esquerda calculada pelo software foi de

$$T_e = 451,96 \text{ K}$$

3 Análise e Discussão dos Resultados

Figura 2.2: Evolução da temperatura T_e calculada pelo software a cada iteração do loop.



Fonte: elaboração própria.