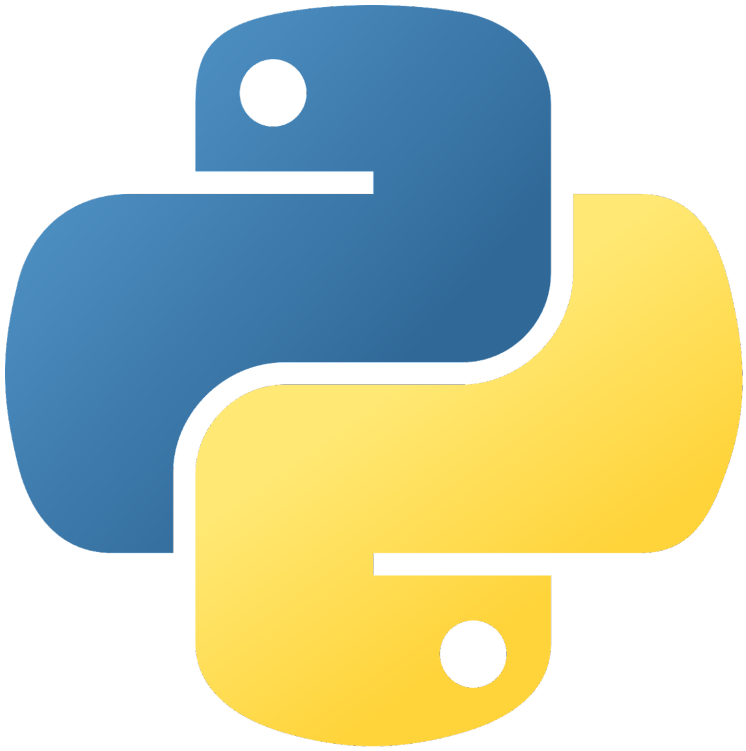
Manual - Resfriamento de chapa



Desenvolvido por:

Enzo Giovanni Benko	Nº USP: 13677292	enzobenko@usp.br
Gabriela Kaori T. Ueno	Nº USP: 13862239	gabrielaueno@usp.br
Lucas R. R. de Almeida	Nº USP: 12686190	lribeirodealmeida@usp.br
Matheus Capelin	Nº USP: 13835990	matheus.capelin@usp.br
Pedro G. P. de Moraes	Nº USP: 13677368	pedrogabrielmoraes@usp.br
Rafael Cândido Reis	Nº USP: 13713361	rafacandidoreis@usp.br

Disciplina:
Computação Científica em Python (LOM3260)

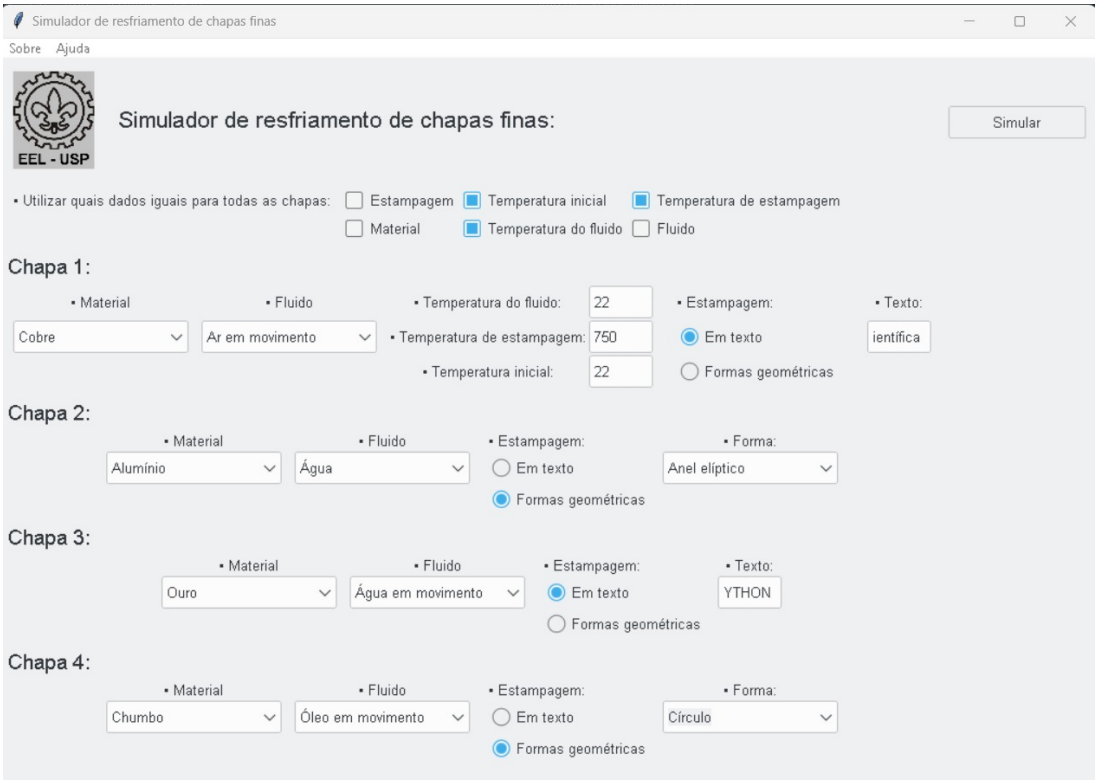
Docente:
Luiz Tadeu Fernandes Eleno

1. Introdução

Os códigos em Python destacados neste manual têm como finalidade simular o resfriamento de chapas metálicas por convecção. Eles foram criados para que o usuário, que fará contato com o programa por meio de uma interface gráfica, analise o resfriamento de chapas de diferentes materiais. Vale ressaltar que, para facilitar o desenvolvimento do programa, foi definida uma chapa ideal com apenas a face frontal em contato com fluidos e objetos de transferência de calor. O programa utiliza o método das diferenças finitas e apresenta condições periódicas de contorno da chapa.

Os materiais considerados para simulação são os seguintes: bronze, ferro, aço, alumínio, prata, ouro, cobre e chumbo. Quanto aos fluidos, foram adotados água, ar e óleo (todos no modo estático ou dinâmico). O primeiro contato que o usuário terá com as funções concebidas pelo código será por meio de uma interface de controle dos dados da simulação. A figura 1 destaca a janela gráfica que aparecerá inicialmente.

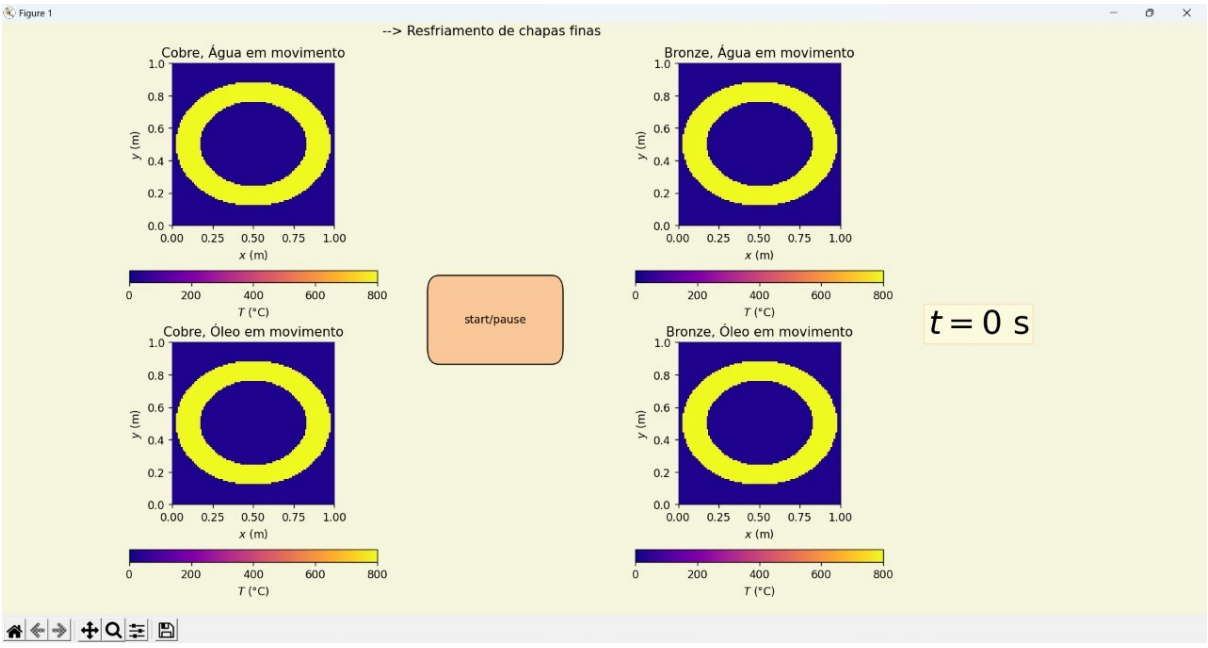
Figura 1 - Janela gráfica com caixas de controle da simulação de resfriamento



É possível avaliar o resfriamento de quatro tipos de chapa ao mesmo tempo. O usuário pode escolher a estampagem, o material, o fluido e a temperatura inicial da chapa, do fluido e da

estampagem (em graus Celsius). A estampagem pode ser gerada em forma geométrica ou texto, e o fluido escolhido pode ser estático ou dinâmico. Após determinar as condições de resfriamento, basta clicar no botão "simular" localizado no canto superior direito da janela e observar a simulação. A figura 2 indica um exemplo de simulação entre quatro situações de resfriamento diferentes.

Figura 2 - Simulação de resfriamento de chapas metálicas em Python



O botão "start/pause" localizado na parte central faz com que a simulação pare e retorne novamente de acordo com os critérios do usuário. O tempo de resfriamento é contabilizado em segundos e também é apresentado na janela de simulação.

2. O código e seu funcionamento

2.1 Interface Gráfica

O código da interface está presente no *start.py* e foi projetado com o módulo tkinter que é nativo do Python, por isso não tem necessidade de download adicional, e o PIL. A interface utiliza a geometria de grid e possui em sua janela principal:

- A área inicial, com o título, ícone e opções de escolher deixar constantes as variáveis entre as chapas.

- As áreas designadas a cada gráfico de chapa, para escolher as variáveis e condições de cada chapa a ser simulada.
- A barra de menu, com as abas sobre, integrantes e ajuda.

2.2 - Painel gráfico e Classes importantes

→ Classes do código

- InitialCondition: esta classe gera a estampa que será usada para esquentar a chapa. Aqui temos 2 funções:
    1. gera_caracteres: irá gerar uma string como estampagem;

```python
class InitialCondition:

    def __init__(self):
        pass

    def gera_caracteres(self, text='COOL', nx1=101, ny1=101, fontsize=24):

        backcolor = 0
        textcolor = 1

        font = ImageFont.truetype("FreeSerif.ttf", fontsize, encoding="unic")
        img = Image.new('L', (nx1, ny1), color=backcolor)

        d = ImageDraw.Draw(img)
        d.text((10, 150), text, fill=textcolor, font=font)

        return np.array(img)[::-1, :]
```

    2. gera_forma: irá gerar uma forma geométrica como estampagem;

```python
def gera_forma(self, forma='Anel circular', nx=101, ny=101, dx=0.01, dy=0.01, Tmax=800, T0=np.zeros([101,101])):

    if forma == "Anel circular":
        for i in range(nx):
            for j in range(ny):
                if ( ( (i*dx-0.5)**2+(j*dy-0.5)**2 <= 0.1)
                    & ((i*dx-0.5)**2+(j*dy-0.5)**2 >= .05) ):
                    T0[i,j] = Tmax
        pass
    elif forma == "Anel elíptico":
        for i in range(nx):
            for j in range(ny):
                if ( ( ((i*dx-0.5)/1.2)**2+((j*dy-0.5)/1.5)**2 <= 0.1)
                    & (((i*dx-0.5)/1.2)**2+((j*dy-0.5)/1.5)**2 >= .05) ):
                    T0[i,j] = Tmax
        pass
    elif forma == "Círculo":
        for i in range(nx):
            for j in range(ny):
                if ((i*dx-0.5)**2 + (j*dx-0.5)**2 <= 0.1):
                    T0[i, j] = Tmax
        pass
    elif forma =="Furo circular":
        for i in range(nx):
            for j in range(ny):
                if ((i*dx-0.5)**2 + (j*dy-0.5)**2 >= .05):
                    T0[i, j] = Tmax
        pass
    else:
        print('Error')
        pass

    return T0
```

Ambas podem ser escolhidas como visto no código e na interface.

- geometry: Aqui é dado o tamanho da chapa em 3 dimensões.

```python
class geometry:

    def __init__(self, Lx=1, Ly=1, Lz=.05):
        self.Lx, self.Ly, self.Lz = Lx, Ly, Lz
```

- mesh: Esta classe irá nos fornecer o tamanho dos pixels no plot do gráfico.

```python
class mesh:

    def __init__(self, nx=101, ny=101):
        self.nx, self.ny = nx, ny
```

- convection: armazena a temperatura de resfriamento que pode ser escolhida na interface (temperatura do fluido).

```python
class convection:

    def __init__(self, Trefr=0):
        self.Trefr = Trefr
```

- material: declaramos as variáveis dos materiais.
- material_database: importamos os dados dos materiais armazenados em um arquivo csv.

```python
class material:

    def __init__(self, k=398, rho=8970, cp=380, name='material'):
        self.name = name
        self.k, self.rho, self.cp = k, rho, cp
        self.a = k / rho / cp

class material_database:

    def __init__(self, xls='materiais_edited.xls'):
        self.dataframe = pandas.read_excel(xls, index_col='Material')

    def select(self, name):
        try:
            k = self.dataframe.loc[name, 'k']
            rho = self.dataframe.loc[name, 'rho']
            cp = self.dataframe.loc[name, 'cp']
            return material(k, rho, cp, name)

        except KeyError as key:
            print(f'*** KeyError: unknown material ({key}) ***')
            raise
```

- fluid: declaramos as variáveis dos fluidos.
- fluid_database: importamos os dados dos fluidos armazenados em um arquivo csv.

```python
class fluid:

    def __init__(self, h=2000, name='fluid'):
        self.name = name
        self.h = h

class fluid_database:

    def __init__(self, xls='materiais_edited.xls'):
        self.dataframe = pandas.read_excel(xls, index_col='Fluid')

    def select(self, name):
        try:
            h = self.dataframe.loc[name, 'h']
            return fluid(h, name)

        except KeyError as key:
            print(f'*** KeyError: unknown fluid ({key}) ***')
            raise
```

- stamping: Esta classe faz uso da classe "InitialCondition", pois ela fará com que a escolha das funções possa ocorrer (estampagem por texto ou forma geométrica).

```python
class stamping:

    def __init__(self, Tstamp=800, Tinitial=100, type=1, text='', fontsize=4, nx1=303, ny1=303, forma='', nx2=101, ny2=101, dx=0.01, dy=0.01, T0=np.zeros([101, 101]
        self.Tstamp = Tstamp
        self.Ti = Tinitial
        self.type = type
        if type == 1:
            self.T0  = np.ones([nx1,ny1]) * self.Ti
            self.T0 += text_image.InitialCondition.gera_caracteres(self, text, nx1, ny1, fontsize) * self.Tstamp

        else:
            self.T0  = np.ones([nx2,ny2]) * self.Ti
            self.T0 += text_image.InitialCondition.gera_forma(self, forma, nx2, ny2, dx, dy, Tstamp, T0) * self.Tstamp
```

- Difffin: Difffin faz uso de todas as classes apresentadas anteriormente. Esta é responsável por articular todas as variáveis apresentadas até então (função __init__) e também efetua os cálculos que serão usados para animação gráfica (função evolve).

```python
class DiffFin:

    def __init__(self, geometry, material, fluid, stamping, convection, mesh=mesh(), dt=.01):
        self.geometry  = geometry
        self.material  = material
        self.fluid = fluid
        self.stamping  = stamping
        self.convection = convection
        self.mesh       = mesh

        dx = geometry.Lx / (mesh.nx - 1)
        dy = geometry.Ly / (mesh.ny - 1)
        dx2 = dx ** 2
        dy2 = dy ** 2
        self.dt = dt

        self.Fo = material.a * dt / geometry.Lz ** 2
        self.Bi = fluid.h * geometry.Lz / material.k

        self.Fox = material.a * dt / dx2
        self.Foy = material.a * dt / dy2

        self.T = np.copy(stamping.T0)

        # variável auxiliar para CC periódicas
        shx, shy = self.T.shape
        self.v = np.zeros((shx+2, shy+2))

    def evolve(self):

        ### condições de contorno periódicas
        self.v[1:-1, 1:-1] = self.T[:, :]
        self.v[0, 1:-1] = self.T[-1, :]
        self.v[-1, 1:-1] = self.T[0, :]
        self.v[1:-1, 0] = self.T[:, -1]
        self.v[1:-1, -1] = self.T[:, 0]

        # diferenças finitas
        self.v[1:-1, 1:-1] += \
            + self.Fox * (self.v[2:, 1:-1] - 2 * self.v[1:-1, 1:-1] + self.v[:-2, 1:-1]) \
            + self.Foy * (self.v[1:-1, 2:] - 2 * self.v[1:-1, 1:-1] + self.v[1:-1, :-2]) \
            - self.Bi * self.Fo * (self.v[1:-1, 1:-1] - self.convection.Trefr)

        self.T = self.v[1:-1, 1:-1]
```

Agora vejamos o código do painel gráfico.

No início, temos a função simula(), que todo o processo do painel irá ocorrer quando essa função é chamada no código da interface (arquivo *start.py*). Aqui também é chamado os materiais e fluidos para este código.

```python
# --- PAINEL ---
def simula(m1='Cobre',m2='Ouro',m3='Ferro',m4='Bronze',f1='Ar',f2='Ar em movimento',f3='Água',f4='Água em movimento',ti1=100,ti2=100,ti3=100,ti4=100,ts1=800,ts2=80

    # parâmetros da geometria da chapa
    chapa = simulation.geometry(Lx=1, Ly=1, Lz=5e-3)

    # inicializando a base de dados (csv)
    db = select_material.material_database('materiais_edited.xls')
    db2 = select_material.fluid_database('materiais_edited.xls')

    # chamando o conteúdo do arquivo.csv para o python
    Material1 = db.select(m1)
    Material2 = db.select(m2)
    Material3 = db.select(m3)
    Material4 = db.select(m4)

    Fluido1 = db2.select(f1)
    Fluido2 = db2.select(f2)
    Fluido3 = db2.select(f3)
    Fluido4 = db2.select(f4)
```

Em seguida, temos a declaração das possíveis estampas que podem ser escolhidas na interface junto da temperatura de resfriamento do fluido. Além disso, temos as simulações, que terão como

entrada a chapa (igual para todas), material, fluido, estampa e temperatura de resfriamento (entradas do usuário na interface, podendo ser diferentes combinações de escolhas e valores..

```
    # parâmetros do processo de estampagem
    if vs1 == 1:
        estampa1 = simulation.stamping(Tinitial=ti1, Tstamp=ts1, type=vs1, text=s1, fontsize=90)
    else:
        estampa1 = simulation.stamping(Tinitial=ti1, Tstamp=ts1, type=vs1, forma=s1)
    if vs2 == 1:
        estampa2 = simulation.stamping(Tinitial=ti2, Tstamp=ts2, type=vs2, text=s2, fontsize=90)
    else:
        estampa2 = simulation.stamping(Tinitial=ti2, Tstamp=ts2, type=vs2, forma=s2)
    if vs3 == 1:
        estampa3 = simulation.stamping(Tinitial=ti3, Tstamp=ts3, type=vs3, text=s3, fontsize=90)
    else:
        estampa3 = simulation.stamping(Tinitial=ti3, Tstamp=ts3, type=vs3, forma=s3)
    if vs4 == 1:
        estampa4 = simulation.stamping(Tinitial=ti4, Tstamp=ts4, type=vs4, text=s4, fontsize=90)
    else:
        estampa4 = simulation.stamping(Tinitial=ti4, Tstamp=ts4, type=vs4, forma=s4)

    # definindo condição de resfriamento/aquecimento
    conv1 = simulation.convection(Trefr=tf1)
    conv2 = simulation.convection(Trefr=tf2)
    conv3 = simulation.convection(Trefr=tf3)
    conv4 = simulation.convection(Trefr=tf4)

    # inicializando as simulações
    simulacao1 = simulation.DiffFin(chapa, Material1, Fluido1, estampa1, conv1)   # ARGUMENTOS:
    simulacao2 = simulation.DiffFin(chapa, Material2,  Fluido2,  estampa2, conv2)  #  1. geommetria
    simulacao3 = simulation.DiffFin(chapa, Material3, Fluido3, estampa3, conv3)   #  2. material 3. fluido
    simulacao4 = simulation.DiffFin(chapa, Material4, Fluido4,  estampa4, conv4) #  4. estampagem 5. convecção
```

Na sequência, temos o início do código do gráfico. É interessante ressaltar a função painel() que prepara as condições da animação e alguns utensílios que melhoram a visualização como o barra inferior que nos mostra o nível de calor que cada cor representa.

```
    # Gráficos
    fig = plt.figure(facecolor='#f5f5dd', constrained_layout=True)
    plt.get_current_fig_manager().window.state('zoomed')

    plt.suptitle(x=0.325, t='--> Resfriamento de chapas finas')

    # função para criar um painel no eixo ax
    def painel(simulacao, ax, titulo=''):

        ax.set_title(titulo)
        ax.set_xlabel(f'$x$ (m)')
        ax.set_ylabel(f'$y$ (m)')

        im = ax.imshow(simulacao.T, vmin=0, vmax=simulacao.stamping.Tstamp, origin='lower', cmap='plasma', extent=(0, simulacao.geometry.Lx, 0, simulacao.geomet
        plt.colorbar(im, ax=ax, location='bottom', orientation='horizontal', label='$T$ (°C)', shrink=.5)

        return im

    ax1 = fig.add_subplot(3, 3, 1)
    ax2 = fig.add_subplot(3, 3, 4)
    ax3 = fig.add_subplot(3, 3, 2)
    ax4 = fig.add_subplot(3, 3, 5)

    im1 = painel(simulacao1, ax1, f'{simulacao1.material.name}, {simulacao1.fluid.name}')
    im2 = painel(simulacao2, ax2, f'{simulacao2.material.name}, {simulacao2.fluid.name}')
    im3 = painel(simulacao3, ax3, f'{simulacao3.material.name}, {simulacao3.fluid.name}')
    im4 = painel(simulacao4, ax4, f'{simulacao4.material.name}, {simulacao4.fluid.name}')

    # Função para pausar com o mouse
    def onClick(event):
        global pause
        pause ^= True
```

Para finalizar, temos o final da nossa função simula(), em que foi criado o cronômetro e o botão para comporem o painel gráfico. Ademais, temos a função animate(), que executará a animação do resfriamento de chapas que podemos observar nos gráficos.

```
    # cronômetro no painel
    axes2 = plt.axes([0.6, 0.56, 0.2, 0.1])
    axes2.axis('off')
    time_label = axes2.text(0.075, 0.2, f'$t = 0$ s', fontsize=30)
    time_label.set_bbox(dict(facecolor='lightyellow', alpha=0.5, edgecolor='#fac898'))

    # função de animação
    def animate(i):
        global pause, n
        if not pause:
            for s, im in zip([simulacao1, simulacao2, simulacao3, simulacao4], [im1, im2, im3, im4]):
                s.evolve()
                im.set_array(s.T)
            time_label.set_text(f'$t = {n * simulacao1.dt: .2f}$ s')
            n += 1
        return  im1, im2, im3, im4, time_label

    delay=0.1
    anim = animation.FuncAnimation(fig, animate, interval=delay, blit=True)

    # botão start/pause customizado
    axes = plt.axes([0.29, 0.56, 0.075, 0.075])
    axes.set_frame_on(False)

    bpause = Button(axes, 'start/pause', color="#fac898")
    bpause.on_clicked(onClick)

    fancybox = mpatches.FancyBboxPatch((0,0), 1,1,
                                       edgecolor="black",
                                       facecolor="#fac898",
                                       boxstyle="round,pad=0.1",
                                       mutation_aspect=3,
                                       transform=axes.transAxes, clip_on=False)
    axes.add_patch(fancybox)

    plt.show()
```

Aqui, acabamos as expecificações.

3. Bibliotecas registradas

As bibliotecas registradas utilizadas no código são: *Numpy, Matplotlib, PIL, Pandas e Tkinter*.

REFERÊNCIAS

CAETANO, Mario. **Calor específico**. Disponível em: https://www.ctborracha.com/borracha-sintese-historica/propriedades-das-borrachas-vulcanizadas/propriedades-fisicas/propriedades-termicas/calor-especifico/. Acesso em: 30 nov. 2022.

CARVALHO, Júlio C. de. **Densidade - Relação entre massa e volume**. Disponível em: https://educacao.uol.com.br/disciplinas/quimica/densidade-relacao-entre-massa-e-volume.htm#:~:text=Projéteis comuns%2C com chumbo (densidade,%2C1g%2Fcm3). Acesso em: 30 nov. 2022.

DONOSO, José. **Calor, energia e transferência de calor**. Disponível em: https://www.ifsc.usp.br/~donoso/fisica_arquitetura/Transferencia_de_Calor.pdf. Acesso em: 30 nov. 2022.

Engineers Edge. **Convective Heat Transfer Coefficients Table Chart**. Disponível em: https://www.engineersedge.com/heat_transfer/convective_heat_transfer_coefficients__13378.htm. Acesso em: 12 nov. 2022.

PRASS, Alberto. **Calor Específico de uma Substância**. Disponível em: https://www.fisica.net/constantes/calor-especifico-c.php. Acesso em: 30 nov. 2022.

RASTEIRO, Maria. **Transferência de calor**. Disponível em: http://labvirtual.eq.uc.pt/siteJoomla/index.php?option=com_content&task=view&id=248&Itemid=422#3. Acesso em: 30 nov. 2022.

SOLIDWORKS. **Coeficiente de calor de convecção**. Disponível em: https://help.solidworks.com/2013/portuguese-brazilian/SolidWorks/cworks/c_Convection_Heat_Coefficient.htm?format=P&value=. Acesso em: 30 nov. 2022.