第三章

3-10

采用 Python 进行计算机程序编写，计算程序如下：

```
import numpy as np
import matplotlib.pyplot as plt
import os

# 使用 Times New Roman 作为 matplotlib 全局字体
plt.rcParams["font.family"] = "serif"
plt.rcParams["font.serif"] = ["Times New Roman"]
plt.rcParams["mathtext.fontset"] = "stix"

class PRFluid:
    def __init__(self, Tc, Pc, omega, M):
        self.Tc = Tc # K
        self.Pc = Pc # Pa
        self.omega = omega # 无量纲
        self.M = M # kg/mol

    R = 8.314462618 # J/(mol*K)

    # 计算a和b
    def params(self, T):
        kappa = 0.37464 + 1.54226 * self.omega - 0.26992 * self.omega**2
        Tr = T / self.Tc
        alpha = (1 + kappa * (1 - Tr**0.5)) ** 2
        a = 0.45724 * self.R**2 * self.Tc**2 / self.Pc * alpha
        b = 0.07780 * self.R * self.Tc / self.Pc
        return a, b

    # 计算A和B
    def AB(self, T, P):
        a, b = self.params(T)
        A = a * P / (self.R * T) ** 2
        B = b * P / (self.R * T)
        return A, B

    # 计算C2, C1, C0
```

```
def C(self, T, P):
    A, B = self.AB(T, P)
    C2 = -(1 - B)
    C1 = A - 3 * B**2 - 2 * B
    C0 = -(A * B - B**2 - B**3)
    return C2, C1, C0

# 计算压缩因子Z
# 液相
def Zl(self, T, P):
    C2, C1, C0 = self.C(T, P)
    # 牛顿法求解Z
    Zl = 0.001 # 初始猜测值
    for _ in range(100):
        f = Zl**3 + C2 * Zl**2 + C1 * Zl + C0
        df = 3 * Zl**2 + 2 * C2 * Zl + C1
        Zl_new = Zl - f / df
        if abs(Zl_new - Zl) < 1e-6:
            break
        Zl = Zl_new
    return Zl

# 气相
def Zg(self, T, P):
    C2, C1, C0 = self.C(T, P)
    # 牛顿法求解Z
    Zg = 1.0 # 初始猜测值
    for _ in range(100):
        f = Zg**3 + C2 * Zg**2 + C1 * Zg + C0
        df = 3 * Zg**2 + 2 * C2 * Zg + C1
        Zg_new = Zg - f / df
        if abs(Zg_new - Zg) < 1e-6:
            break
        Zg = Zg_new
    return Zg

# 计算比体积v
# 液相
def vl(self, T, P):
    Zl = self.Zl(T, P)
    vl = Zl * self.R * T / (P * self.M)
    return vl

# 气相
def vg(self, T, P):
    Zg = self.Zg(T, P)
    vg = Zg * self.R * T / (P * self.M)
    return vg

# 画v-T图
def plot_Tv(
    self,
    fluid_name,
    P,
    Tsat,
    T_min,
    T_max,
    nT=220,
    savepath=None,
    dpi=300,
    bbox_inches="tight",
    transparent=False,
    close_fig=False,
):
    T_grid = np.linspace(T_min, T_max, nT)
    v_grid = np.empty_like(T_grid)
    for i, T in enumerate(T_grid):
        if T < Tsat:
            v_grid[i] = self.vl(T, P)
        elif T > Tsat:
            v_grid[i] = self.vg(T, P)
        else:
            v_grid[i] = 0.5 * (self.vl(T, P) + self.vg(T, P))
    fig, ax = plt.subplots()
    # 主曲线
    ax.plot(v_grid, T_grid, linewidth=2, label=fluid_name)
    xmin, xmax = np.nanmin(v_grid), np.nanmax(v_grid)
    # Tsat 虚线
    ax.hlines(Tsat, xmin, xmax, linestyles="--", label=r"$T_{\mathrm{sat}}$"
        )
```

```
# 用纵轴刻度标注 Tsat
yt = list(ax.get_yticks())
# 如果 Tsat 不在当前刻度中，加入并排序
if not any(abs(t - Tsat) < 1e-8 for t in yt):
    yt.append(Tsat)
yt = np.array(sorted(yt))
# 生成刻度标签：对 Tsat 使用仅数值标签（两位小数），其它刻度保留数字格式
    （根据范围选择小数位）
deltaT = T_grid.max() - T_grid.min()
labels = []
for t in yt:
    if abs(t - Tsat) < 1e-8 or abs(t - Tsat) < 1e-6 * max(1.0, deltaT):
        labels.append(f"{Tsat:.2f}")
    else:
        # 根据温度范围决定格式，避免过多小数
        if deltaT > 50:
            labels.append(f"{t:.0f}")
        else:
            labels.append(f"{t:.2f}")
ax.set_yticks(yt)
ax.set_yticklabels(labels)
# 轴标签
ax.set_xlabel(r"$v$ (m³/kg)")
ax.set_ylabel(r"$T$ (K)")
# 标题
ax.set_title(f"{fluid_name} $v$-$T$ at $P$ = {P/1e6:.3f} MPa")
ax.grid(True)
ax.set_xscale("log") # 使用对数刻度
ax.legend(loc="upper left", frameon=True, fancybox=True, framealpha=0.9)
# 默认保存路径为脚本同目录下的 fig 文件夹（当 savepath 为 None 时）
if savepath is None:
    base_dir = os.path.dirname(os.path.abspath(__file__))
    fig_dir = os.path.join(base_dir, "fig")
    os.makedirs(fig_dir, exist_ok=True)
    # 简单替换文件名中的非法字符
    safe_name = "".join(
        c if c.isalnum() or c in ("_", "-") else "_" for c in fluid_name
    )
    filename = f"{safe_name}_v-T_P{P/1e6:.3f}MPa.png"
    savepath = os.path.join(fig_dir, filename)
```

```
        # 保存图像（如果提供或已生成 savepath）
        if savepath:
            dirname = os.path.dirname(savepath)
            if dirname and not os.path.exists(dirname):
                os.makedirs(dirname, exist_ok=True)
            fig.savefig(
                savepath, dpi=dpi, bbox_inches=bbox_inches, transparent=
                    transparent
            )
            if close_fig:
                plt.close(fig)

        return fig

R290 = PRFluid(369.89, 4.2512e6, 0.1521, 44.096 / 1000) # kg/mol
R290.plot_Tv("R290", 1.4e6, 317.86, 200, 450, savepath="R290_v-T.png")

R600a = PRFluid(407.81, 3.629e6, 0.184, 58.122 / 1000) # kg/mol
R600a.plot_Tv("R600a", 0.6e6, 314.12, 200, 450, savepath="R600a_v-T.png")
```

3-13

3-15

第四章

4-13

采用 Python 进行计算机程序编写，计算程序如下：

4-15