parameters

real global p\_vapour (range= <-1e5,>) {Pa}; // vapour presure

real global rho (range= <0,>) {kg/m3}; // mass density

real Q\_nom (range=<0,>) = 2.0e-5 {m3/s}; // nominal flow at nominal pressure drop

real Q\_leak (range=<0,>) = 5e-8 {m3/s}; // leakage flow at nominal pressure drop

real p\_nom (range=<0,>) = 5e5 {Pa}; // nominal pressure drop

variables

real dp; // pressure drop over the orifice

real hidden pa\_lim; // pressure A-side with minimum limited to vaccuum

real hidden pb\_lim; // pressure B-side with minimum limited to vaccuum

real Cd {}; // discharge coefficient;

real A {m2}; // valve opening area at nominal flow and nominal pressure

real GLeak; // conductance of laminar leakage flow

equations

// parameters

Cd = 0.6;

A = Q\_nom / ( Cd\*sqrt((2 / rho)\*p\_nom) );

GLeak = Q\_leak/p\_nom;

// limit pressure

pa\_lim = if pa.p < p\_vapour then p\_vapour else pa.p end;

pb\_lim = if pb.p < p\_vapour then p\_vapour else pb.p end;

// flow equations

dp = pa\_lim - pb\_lim;

pa.phi = sign(dp) \* Cd \* A \* sqrt( (2/rho) \* abs(dp) ) + GLeak \* dp;

pb.phi = pa.phi;

为了将你的\*\*节流阀（throttle valve）\*\*作为 **BondGraphTools 的复合子模型** 进行构建、封装并保存到“library”中供后续调用，可以按以下思路实现：

## ✅ 一、创建复合子模型（Python 函数封装）

我们将你的节流阀建模逻辑用一个函数封装成复合子模型：

import BondGraphTools as bgt

import numpy as np

def build\_throttle\_valve(name="ThrottleValve", params=None):

"""

构建节流阀子系统，支持参数配置

"""

# 默认参数

default\_params = {

"Cd": 0.6,

"rho": 1000, # kg/m^3

"Q\_nom": 2.0e-5, # m^3/s

"Q\_leak": 5e-8, # m^3/s

"p\_nom": 5e5, # Pa

"p\_vapour": 0 # Pa (最小压力)

}

# 使用默认参数或覆盖

if params:

default\_params.update(params)

p = default\_params

# 计算阀口面积和泄露导通

A = p["Q\_nom"] / (p["Cd"] \* np.sqrt((2 / p["rho"]) \* p["p\_nom"]))

G\_leak = p["Q\_leak"] / p["p\_nom"]

# 创建复合系统

valve = bgt.new(name=name)

# 构建元件

R\_throttle = bgt.new("R", value=1/(p["Cd"] \* A), name="R\_throttle")

R\_leak = bgt.new("R", value=1/G\_leak, name="R\_leak")

zero = bgt.new("0", name="Node")

# 添加元件

bgt.add(valve, R\_throttle, R\_leak, zero)

# 连接 R\_throttle 和 R\_leak 到共节点

bgt.connect(R\_throttle, zero)

bgt.connect(R\_leak, zero)

return valve

## ✅ 二、保存到本地“library”模块

你可以在你的项目中建一个 bondgraph\_components/ 文件夹，创建一个 valves.py 文件：

# 文件: bondgraph\_components/valves.py

from BondGraphTools import new, add, connect

import numpy as np

def build\_throttle\_valve(name="ThrottleValve", params=None):

...

# 将上面 build\_throttle\_valve 函数的完整代码粘贴在此

之后即可这样调用：

from bondgraph\_components.valves import build\_throttle\_valve

valve = build\_throttle\_valve(params={"Q\_nom": 1e-4})

## ✅ 三、添加到主模型中

system = bgt.new(name="MainSystem")

valve1 = build\_throttle\_valve("Valve1")

valve2 = build\_throttle\_valve("Valve2", params={"Q\_nom": 1e-4})

bgt.add(system, valve1, valve2)

# 如有需要再连接 valve1 和 valve2

## 🧩 拓展建议：

每个子模型可以设计标准输入输出端口（如端口 0/1），便于统一接入；

可定义统一的 ComponentLibrary 类来注册、查找、调用模型；

后续可加入 json/yaml 配置生成子模型。

需要我帮你生成一个完整的 valves.py 模板？或者展示如何创建一个标准组件库系统？你可以点击这里快速查看 [复合子模型库模板](https://chatgpt.com/c/f)。