

Vapor box designer!

Constants and vapor pressure formulae

Script characters are for quantities with dimensions. **DoubleStruck** characters are for units themselves.

```
In[1]:= Q[a_, b_] := Quantity[a, b];  
Q[a_] := Quantity[a];  
uc[a_, b_] := UnitConvert[a, b];  
uc[a_] := UnitConvert[a];  
ucSI[a_] := UnitConvert[a, "SI"];  
kg = Q["Kilograms"];  
m = Q["Meters"];  
K = Q["Kelvins"];  
J = Q["Joules"];  
s = Q["Seconds"];  
Pa = Q["Pascals"];  
W = Q["Watts"];
```

```
In[13]:= celsiusToKelvin = Q[0., "DegreesCelsius"] // UnitConvert // QuantityMagnitude;  
(* 273.15 *)
```

Lithium

```
In[14]:= << lithiumDataARK`
```

Other materials

```
In[15]:= << materialDataARK`
```

Geometry

Assumes that only the walls and base are emitting/absorbing surfaces, (since the baffles are reflecting).
Base is only for the first chamber.

```
In[16]:= calculateGeometryCylinder[boxLength_, boxDiameter_, nozzleDiameter_, nBoxes_] :=
  Block[{Awall, Anozz, ca, nozzleRadius, boxRadius},
    ca = ConstantArray[1, nBoxes];
    (*hack to also accept differing slotWidthi, etc*)
    nozzleRadius = nozzleDiameter/2;
    boxRadius = boxDiameter/2;
    Awall = 2  $\pi$  boxRadius boxLength ca;
    Awall[[1]] +=  $\pi$  boxRadius2;
    Anozz =  $\pi$  nozzleRadius2 ca;
    {Awall, Anozz}
  ]
```

Calculates the full area of the cylinder, minus the first hole.

```
In[17]:= calculateFullAreaCylinders[boxLength_, boxDiameter_, nozzleDiameter_, nBoxes_] :=
  Block[{Awall, Anozz, ca, nozzleRadius, boxRadius},
    ca = ConstantArray[1, nBoxes];
    (*hack to also accept differing slotWidthi, etc*)
    nozzleRadius = nozzleDiameter/2;
    boxRadius = boxDiameter/2;
    Anozz =  $\pi$  nozzleRadius2 ca;
    Awall = 2  $\pi$  boxRadius boxLength ca + 2  $\pi$  boxRadius2 ca - 2 Anozz;
    Awall[[1]] += (Anozz)[[1]];
    {Awall, Anozz}
  ]
```

A combination of the above functions for a single-box. Needs some work as for I/O: what's an array vs not.

```
In[18]:= geometry[length_, diameter_, nozzleRatio_, nBoxes_] :=
  Block[{AwallEvap, nozzleDiameter, AwallRadiate, Anozz},
    nozzleDiameter = diameter/nozzleRatio;
    {AwallEvap, Anozz} =
      calculateGeometryCylinder[length, diameter, nozzleDiameter, nBoxes];
    {AwallRadiate, Anozz} = calculateFullAreaCylinders[
      length, diameter, nozzleDiameter, nBoxes];
    <|"l" → length, "d" → diameter, "AWet" → AwallEvap, "ATot" → AwallRadiate,
      "Anoz" → Anozz, "dnoz" → nozzleDiameter, "nBoxes" → nBoxes|>
  ]
```

Can geometry

```
In[19]:= shellVolume[geom_, thickness_] := geom["ATot"] thickness
shellMass[geom_, thickness_, material_] :=
  materialLib[material]["ρ"] shellVolume[geom, thickness]
shellThermalMass[geom_, thickness_, material_] :=
  materialLib[material]["s"] shellVolume[geom, thickness]
```

Radiation calculation

Gross outgoing radiation (like radiating into a cold void) for a convex object with emissivity ϵ , area A , and temperature \mathcal{T} .

```
In[22]:= radiatedPower $\epsilon A \sigma T^4$ [ $\epsilon$ _,  $\mathcal{A}$ _,  $\mathcal{T}$ _] := Block[{ $\sigma_{sb}$ },
   $\sigma_{sb}$  = Quantity["StefanBoltzmannConstant"];
  UnitConvert[ $\epsilon \sigma_{sb} \mathcal{A} \mathcal{T}^4$ , W]
];
```

Radiation transfer between two concentric cylinders of negligibly different diameters, if they were infinite in length.

```
In[23]:= radiationTransferEfficiencyFactor[geom_, mat1_, mat2_] := Block[{ $\sigma_{sb}$ ,  $\epsilon_1$ ,  $\epsilon_2$ },
   $\sigma_{sb}$  = Quantity["StefanBoltzmannConstant"];
  { $\epsilon_1$ ,  $\epsilon_2$ } = materialLib[#]["ε"] & /@ {mat1, mat2};

  
$$\frac{\sigma_{sb}}{\frac{1}{\epsilon_1} + \frac{1}{\epsilon_2} - 1} \text{geom["ATot"][[1]] // UnitConvert[#, "Watts" "Kelvins"^-4] \&}$$

];
```

Radiation transfer between cylindrical shells as if they were an infinite-length concentric cylinder geometry. This ignores the end faces of the cylinder.

```
In[24]:= concentricCylinderRTEF[length_, r1_, r2_, mat1_, mat2_] := Block[{ $\sigma_{sb}$ , R,  $\epsilon_1$ ,  $\epsilon_2$ },
   $\sigma_{sb}$  = Quantity["StefanBoltzmannConstant"];
  { $\epsilon_1$ ,  $\epsilon_2$ } = materialLib[#]["ε"] & /@ {mat1, mat2};
  R = r2 / r1;

  
$$\frac{R \epsilon_1 \epsilon_2}{\epsilon_1 + R \epsilon_2 - \epsilon_1 \epsilon_2} \sigma_{sb} 2 \pi r1 \text{length} // \text{UnitConvert[#, "Watts" "Kelvins"^-4] \&}$$

];
```