Heating, cooling, and lithium flow during a Magnum-PSI experiment

This code generates Figure 5.

Load some routines for calculating areas, geometries, masses etc of cylindrical cans, as well as some data on Lithium properties, properties of other materials used, and dimensions of the radiant heater used.

In[1]:=
```
NotebookEvaluate[NotebookDirectory[] <> "simpleVBD_ARK.nb"]
<< thermcraftHeatersARK`
```

In[3]:=
```
in = Quantity[1., "Inches"];
cm = Quantity[1., "Centimeters"];
```

Load data on the heater.

In[5]:=
```
heater = thermcraftHeaters["RH292"];
```

A description of the central box and the end boxes.

In[6]:=
```
diameter = UnitConvert[heater["ID"] - 1/16 in * 7, cm];
diameter = 16 cm;
dnoz = 6 cm;
ratio = diameter/dnoz;
length = {15.695 2 cm, 10 cm};
thickness = sheetGageSS["16th"];
```

In[12]:=
```
geom1 = geometry[length[[1]], diameter, ratio, 1];
geom2 = geometry[length, diameter, ratio, 2];
```

Typical temperatures of the central box and end boxes.

In[14]:=
```
Texps = {(650 + celsiusToKelvin) K, (300 + celsiusToKelvin) K};
Texp = Texps[[1]];
```

Data on the reflective heat shield that goes around the heater.

In[16]:=
```
shieldThickness = 0.01 in;
shieldDiameter = heater["OD"] + 2 cm;
```

Data on the surrounding vacuum vessel.

In[18]:=
```
VVDiameter = 50 cm;
TVVWall = 300 K;
```

Interpolation of flow rates from DSMC runs.

Several runs with the DSMC code SPARTA were performed, with the central box at temperatures between 500°C and 650°C, and the end boxes fixed at 300°C. The first two columns give temperatures of the central box and end box, respectively, in °C. The 3rd-6th columns are flows (3) out of the upstream end box to the main chamber, (4) from the central box to the upstream end box, (5) from the central box to the downstream end box, and (6) from the downstream end box to the main chamber.

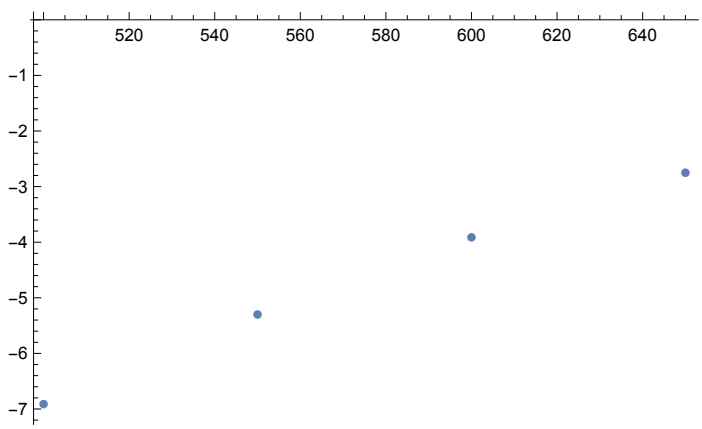
In[20]:=
```
dsmcValues = {
    {500, 300, 5.35 × 10^-5, 0.000502, 0.000494, 1.06 × 10^-5},
    {550, 300, 0.000307, 0.00249, 0.0025, 6.7 × 10^-5},
    {600, 300, 0.00156, 0.00999, 0.01, 0.00054},
    {650, 300, 0.00552, 0.0319, 0.032, 0.0019}} // Transpose;
```

In[21]:=
```
temperatures = dsmcValues[[1]];
fromE = dsmcValues[[3]] + dsmcValues[[6]];
fromC = dsmcValues[[4]] + dsmcValues[[5]];
```

The flows as a function of temperature seem amenable to interpolation in log-y space.

In[94]:=
```
ListPlot[{{temperatures, Log[fromC]}ᵀ}]
```

Out[94]=



Interpolation in log-y space

In[24]:=
```
flow1i =
  Exp[Interpolation[{temperatures, Log[fromC]}ᵀ, InterpolationOrder → 1][#]] &;
flow2i = Exp[Interpolation[{temperatures, Log[fromE]}ᵀ,
      InterpolationOrder → 1][#]] &;
```

In[26]:=
```
massflowrates[T1_, T2_] :=
 Quiet[{flow1i[T1 - celsiusToKelvin], flow2i[T1 - celsiusToKelvin]},
  {InterpolatingFunction::dmval}]
```

In[27]:=
```
vaporMassLoss[TK_, "DSMC"] :=
 Quantity[massflowrates @@ QuantityMagnitude[{TK, 0}], "Grams" / "Seconds"]
vaporHeatLoss[TK_, "DSMC"] := ℓLi vaporMassLoss[TK, "DSMC"]
vaporHeatLoss[TK_Quantity, "DSMC"] :=
 ℓLi vaporMassLoss[QuantityMagnitude[TK], "DSMC"]
```

These ‘rtef’ factors are intermediates in the calculations of power transferred by radiation between the central box, heaters, heat shield, and vacuum vessel. They take into account relative radii of the various cylinders and the emissivities at each outer and inner surface.
“CeramicHeater” calls for the value of a high-emissivity coating, “Uncoated” is lower emissivity, and “SS” refers to the emissivity of stainless steel.

In[30]:=
```
rtef1 = concentricCylinderRTEF[geom1["l"],
   geom1["d"] / 2, heater["ID"] / 2, "CeramicHeater", "CeramicHeater"];
rtef2 = concentricCylinderRTEF[geom1["l"], heater["OD"] / 2,
   shieldDiameter / 2, "CeramicHeaterUncoated", "SS"];
rtef3 = concentricCylinderRTEF[geom1["l"], shieldDiameter / 2,
   VVDiameter / 2, "SS", "SS"];
```

This function finds a steady-state heater temperature required during the experiment, to support the amount of radiation transfer to the central box that equals the latent heat of evaporation of lithium flowing out of that box.

In[33]:=
```
heaterTemperatureExpt[radiationTransferEfficiencyFactor_, geom_, Texp_List] :=
  Print["Error: Texp must be a single real number."];
heaterTemperatureExpt[radiationTransferEfficiencyFactor_, geom_, Texp_] :=
 Block[{heatLost, rtef},
  rtef = radiationTransferEfficiencyFactor;
  heatLost = vaporHeatLoss[Texp, geom][[1]];
  (heatLost + rtef Texp^4)^(1/4) / rtef^(1/4)
 ]
```

Solve for heat shield temperature and total power required for the heater in steady state.

In[35]:=
```
shieldTemperature[𝒯h_, 𝒯VVWall_, rtef2_, rtef3_] := Block[{Tsh, 𝒯sh},
  𝒯sh = Tsh /.
    NSolve[{rtef2 (𝒯h^4 - Tsh^4) == rtef3 (Tsh^4 - 𝒯VVWall^4), Tsh > 0}, Tsh, Reals][[1]]
 ]
𝒯h = heaterTemperatureExpt[rtef1, "DSMC", Texp];
𝒯sh = shieldTemperature[𝒯h, TVVWall, rtef2, rtef3];
```

In[38]:=
```
powerTransferred[rtef_, T1_, T2_] := rtef (T1^4 - T2^4)
steadyStateHeaterPower =
 powerTransferred[rtef1, 𝒯h, Texp] + powerTransferred[rtef2, 𝒯h, 𝒯sh]
```

Out[39]= 2245.6 W

Check that power to wall is << 25 kW / m^2.

25 kW/m^2 is similar to the capability of Magnum-PSI's water-cooled vacuum vessel. Note that this is the power at the diameter of the heat shield itself: it is further spread since the vacuum vessel inner wall diameter is roughly twice the diameter of the heat shield, and because the heat shield radiates forward and backward as well, not only radially.

In[46]:=
```
UnitConvert[powerTransferred[rtef3, τsh, TVVWall] / (π shieldDiameter length[[1]]), "Kilowatts" / "Meters"^2] <
 Quantity[25, "Kilowatts" / "Meters"^2]
```

Out[46]= True

Check steady state heater power

The power required in steady state should be somewhat less than the maximum power rating of the heater, so that the ramp-up time is not excessive.
Note that here, and below, we assume that the heater is powered at 125% of rated power. See the text for an explanation.

In[40]:=
```
power = 2.5 heater["Watts"];
steadyStateHeaterPowerFraction = steadyStateHeaterPower / (power W)
```

Out[41]= 0.767 71

Thermal masses of the three modeled components.

In[42]:=
```
boxS = shellThermalMass[geom2, sheetGage[{"SS", "16th"}], "SS"];
shieldS =
  shellThermalMass[<|"ATot" → shieldDiameter π geom1["l"]|>, shieldThickness, "SS"];
heaterS = 2 heater["HeatCapacity"] (* there are two heaters *);
```

Heating and cooling time of central box

The heating and cooling of the central box during the experiment is calculated in three phases. First, the box heats via thermal radiation from the heaters. Then, the box is additionally heater by the Magnum-PSI plasma beam. Finally, the beam and the heater power are turned off and the box radiatively cools.

The temperatures of the central box, the heater, and the heat shield are solved for during these processes. Each object is treated as having a uniform temperature. Each cylindrical object exchanges thermal radiation with the objects inside and outside it, except there is no treatment of radiation inside the central box, and the vacuum vessel outside the heat shield is treated as having a constant temperature of 300K.

Heat is lost from the inside of the central box due to the latent heat of vaporization of lithium leaving the box. Lithium leaves the box as a function of the box temperature, interpolated from flow rates calculated by DSMC simulations at various temperatures.

Additional effects considered include:
Variation with temperature of the heat capacity of the heater material.
Variation with temperature of the resistance of the heaters, and therefore the powered delivered for a constant voltage.

We stop using Mathematica's Quantity functionality during the integration, since it greatly slows things down.

In[45]:=
```
{rtef1, rtef2, rtef3} = QuantityMagnitude[{rtef1, rtef2, rtef3}];
{boxS, heaterS, shieldS} =
  {boxS[[1]], heaterS[[1]], shieldS} // QuantityMagnitude;
{Tmin, Tmax, tvvwall} = {TVVWall, Texp, TVVWall} // QuantityMagnitude;
vhlif[TK_] := 10^-3 lLi massflowrates[TK, 0][[1]];
(* factor of 10^-3 converts kg^-1 to g^-1 *)
vmlif[TK_] := massflowrates[TK, 0][[1]]
vml2if[TK_] := massflowrates[TK, 0][[2]]
```

Numerically solve the differential equations for Phase 1: heating

In[51]:=
```
endw = 3500;(* warming simulation max time, then gets set to end time  *)
endwpoint = 0;
solw = Quiet[NDSolveValue[{Th[0] == tvvwall, Tb[0] == tvvwall, Ts[0] == tvvwall,
     (power * Piecewise[{{1, Tb[t] < Tmax}}, steadyStateHeaterPowerFraction]) /
       ρElTemperatureFactorKanthal[Th[t] - celsiusToKelvin] ==
      cMulliteTFactor[Th[t]] heaterS Th'[t] + rtef2 (Th[t]^4 - Ts[t]^4) +
       rtef1 (Th[t]^4 - Tb[t]^4), rtef1 (Th[t]^4 - Tb[t]^4) == vhlif[Tb[t]] + boxS Tb'[t],
     rtef2 (Th[t]^4 - Ts[t]^4) == shieldS Ts'[t] + rtef3 (Ts[t]^4 - tvvwall^4),
     WhenEvent[Tb[t] > Tmax - 0.01, endwpoint = t;]}, {Th, Tb, Ts}, {t, 0, endw}]];
```

Record the end temperatures and compute the total lithium emitted from the central box and end boxes during the warmup phase.

In[54]:=
```
warmingEndTemperatures = solw[[#]][endwpoint] & /@ Range[3] - celsiusToKelvin;

integrateOverWarmup[f_] := NIntegrate[f, {t, 0, endwpoint}]
warmupLi = integrateOverWarmup[vmlif[solw[[2]][t]]];
warmup2Li = integrateOverWarmup[vml2if[solw[[2]][t]]];
```

Simple treatment of end boxes

We want to check the assumption that the end boxes remain relatively cool during the experiment, so that they can act as condensing surfaces. The thermal mass of one end box is multiplied by four: twice

because there are two end boxes, and twice more to reflect a thicker material with a larger total thermal mass. It is found that during the heating process of the central box, the end box reaches ~275°C, less than the 300 C assumed in the simulations. The end boxes will continue to heat during the cooling phase due to continued condensation of lithium vapor. If high temperature and subsequent (re)evaporation of lithium is an issue, the end box thermal mass may be increased.

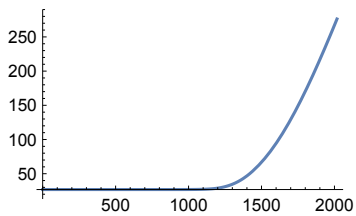
In[58]:=
```
rtefBox2 = 0.85 × 0.3 geom2["ATot"][[2]] Quantity["StefanBoltzmannConstant"] //
    UnitConvert[#, "Watts" / "Kelvins"^4] & // QuantityMagnitude;
```

Net latent heat of deposited lithium vapor.

In[59]:=
```
netHeat[t_] := lLi (vmlif[solw[[2]][t]] - vml2if[solw[[2]][t]]) 10^-3
```

In[60]:=
```
box2S = boxS[[2]] // QuantityMagnitude;
solw2 = Quiet[NDSolveValue[
    {Tb2[0] == tvvwall, netHeat[t] == 4 box2S Tb2'[t] + 1 rtefBox2 (Tb2[t]^4 - tvvwall^4) +
       0 Piecewise[{{500, t > 1500}}, 0]}, Tb2, {t, 0, endwpoint}]];
Plot[solw2[t] - 273.15, {t, 0, endwpoint}, ImageSize → Small, PlotRange → All]
```

Out[62]=



Phase 2: plasma exposure

During this phase, the heaters remain on and the plasma beam is turned on for 10 seconds.

In[63]:=
```
beamPower = 3000 (*watts *);
endb = 10;
phase2StartTemps = #[endwpoint] & /@ solw
```

Out[65]= {1033.1, 923.14, 876.96}

In[66]:=
```
solb =
  Quiet[NDSolveValue[{Th[0] == phase2StartTemps[[1]], Tb[0] == phase2StartTemps[[2]],
     Ts[0] == phase2StartTemps[[3]], power == cMulliteTFactor[Th[t]] heaterS Th'[t] +
       rtef2 (Th[t]^4 - Ts[t]^4) + rtef1 (Th[t]^4 - Tb[t]^4),
     beamPower + rtef1 (Th[t]^4 - Tb[t]^4) == vhlif[Tb[t]] + boxS Tb'[t],
     rtef2 (Th[t]^4 - Ts[t]^4) == shieldS Ts'[t] + rtef3 (Ts[t]^4 - tvvwall^4)},
    {Th, Tb, Ts}, {t, 0, endb}]];
```

Phase 3: cooling

In[67]:=
```
phase3StartTemps = #[endb] & /@ solb
```

Out[67]= {1034.5, 952.94, 877.82}

In[68]:=
```
endc = 5000; (* maximum end time of integration *)
```

In[69]:=
```
solc =
  Quiet[NDSolveValue[{Th[0] == phase3StartTemps[[1]], Tb[0] == phase3StartTemps[[2]],
     Ts[0] == phase3StartTemps[[3]], 0 == cMulliteTFactor[Th[t]] heaterS Th'[t] +
       rtef2 (Th[t]^4 - Ts[t]^4) + rtef1 (Th[t]^4 - Tb[t]^4),
     rtef1 (Th[t]^4 - Tb[t]^4) == vhlif[Tb[t]] + boxS Tb'[t], rtef2 (Th[t]^4 - Ts[t]^4) ==
      shieldS Ts'[t] + rtef3 (Ts[t]^4 - tvvwall^4), WhenEvent[Tb[t] < 700, endcpoint = t;
      "StopIntegration"]}, {Th, Tb, Ts}, {t, 0, endc}]];
```

Calculate lithium flow during the cooling period

In[70]:=
```
integrateOverCooldown[f_] := NIntegrate[f, {t, 0, endcpoint}]
cooldownLi = integrateOverCooldown[vmlif[solc[[2]][t]]]
cooldown2Li = integrateOverCooldown[vml2if[solc[[2]][t]]]
```

Out[71]= 31.736

Out[72]= 3.256

Join together the heating, plasma beam exposure, and cooling simulation results.

In[73]:=
```
warmBeamCool[index_, t_] :=
 Piecewise[{{solw[[index]][t], t < endwpoint}, {solc[[index]][t - endwpoint - endb],
    t > endb + endwpoint}}, solb[[index]][t - endwpoint]]
```

Total quantity of lithium that flows out of the central box.

In[74]:=
```
NIntegrate[vmlif[warmBeamCool[2, t]], {t, 0, 9000}]
```

Out[74]= 61.675

Total quantity of lithium that flows out of the end boxes and into the main chamber.

In[75]:=
```
NIntegrate[vml2if[warmBeamCool[2, t]], {t, 0, 9000}]
```

Out[75]= 6.581 5

Function for multiple plots together in a grid.

Plot the results

In[76]:=
```
plheatbcool =
  Plot[{warmBeamCool[1, t], warmBeamCool[2, t], warmBeamCool[3, t]}, {t, 0, 4750},
   PlotLegends → Placed[{"Heater", "Central Box", "Shield"}, {0.7, 0.3}],
   Frame → {{True, False}, {False, True}},
   FrameTicks → {{{300, 400, 500, 600, 700, 800, 900, 1000}, None}, Automatic},
   FrameLabel → {, "Temperature / K"}, LabelStyle → 12,
   Epilog → Text[Style["a", 15], Scaled[{0.075, 0.9}]]];
plLiEm = Plot[{vmlif[warmBeamCool[2, t]], vml2if[warmBeamCool[2, t]]}, {t, 0, 4750},
   PlotRange → {0, 0.13}, PlotPoints → 500, Frame → {{True, False}, {True, True}},
   FrameTicks → {{Automatic, False}, {{0, 1000, 2000, 3000, 4000}, False}},
   GridLines → None, PlotStyle → {Black, Lighter[Purple]},
   FrameLabel → {"Time / s", "Li vapor flow / (g/s)"},
   PlotLegends → Placed[{"Flow from central box\n to end boxes",
      "Flow from end boxes\n to main chamber"}, {0.72, 0.7}],
   LabelStyle → 12, Epilog → Text[Style["b", 15], Scaled[{0.075, 0.9}]]];
```

Time to warm up

In[78]:=
```
endwpoint
```

Out[78]=
```
2017.3
```

Estimate of the amount of Li emitted during the temperature spike caused by the plasma beam.

In[79]:=
```
NIntegrate[vmlif[warmBeamCool[2, t]], {t, endwpoint, 2068.5}]
```

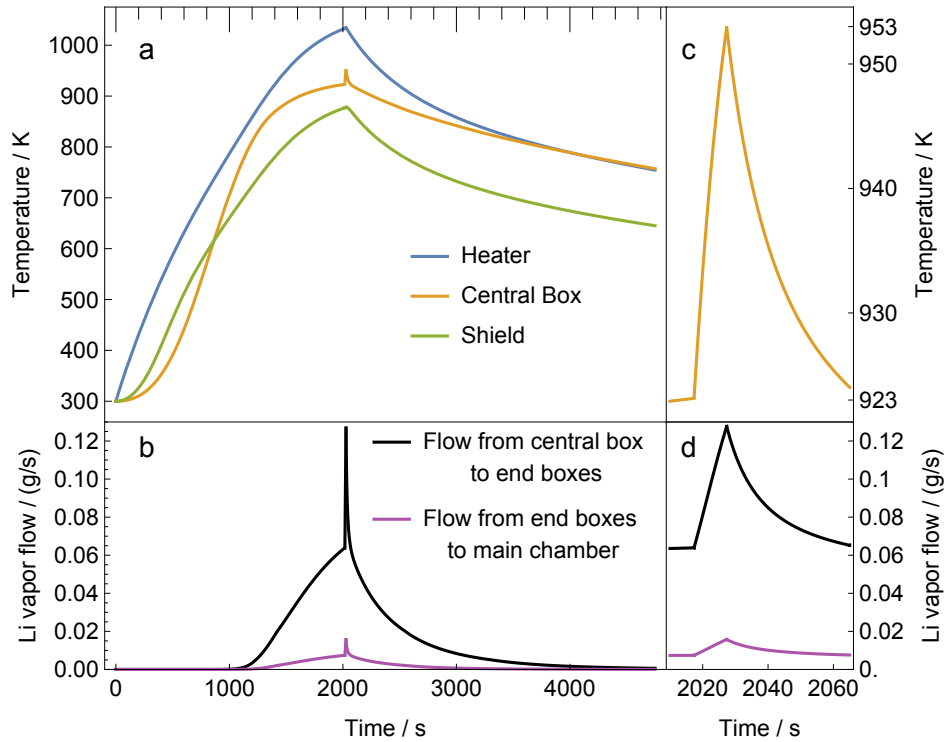
Out[79]=
```
4.290 9
```

Inset pictures for the plot.

In[86]:=
```
plheatbcoolInset = Plot[{, warmBeamCool[2, t],},
   {t, 2010, 2065}, Frame → {{True, True}, {True, True}}, FrameTicks →
    {{None, {800, 850, 900 , 920, 923, 930, 940, 950, 953, 1000}}, {None, Automatic}},
   FrameLabel → {, , , "Temperature / K"}, LabelStyle → 12,
   Epilog → Text[Style["c", 15], Scaled[{0.12, 0.9}]]];
plLiEmInset = Plot[{vmlif[warmBeamCool[2, t]], vml2if[warmBeamCool[2, t]]},
   {t, 2010, 2065}, PlotRange → {0, 0.13},
   PlotPoints → 500, Frame → {{True, True}, {True, True}},
   FrameTicks → {{False, {0.00, 0.02, 0.04, 0.06, 0.08, 0.10, 0.12}},
     {{2000, 2020, 2040, 2060}, False}},
   GridLines → None, PlotStyle → {Black, Lighter[Purple]},
   FrameLabel → {"Time / s", , , "Li vapor flow / (g/s)"},
   LabelStyle → 12, Epilog → Text[Style["d", 15], Scaled[{0.12, 0.9}]]];
```

In[88]:=
```
pg = plotGrid[{{plheatbcool, plheatbcoolInset}, {plLiEm, plLiEmInset}},
  500, 400, {3, 1}, {0.8, 1}, ImagePadding → 55]
```

Out[88]=



Export of data for archiving

```
Figure5DataTopRaw = Table[
   {t, warmBeamCool[1, t], warmBeamCool[2, t], warmBeamCool[3, t]}, {t, 0, 4750}];
extraPoints = {#, warmBeamCool[1, #], warmBeamCool[2, #], warmBeamCool[3, #]} & /@
   {endwpoint, endwpoint + endb};
Figure5DataTop = Sort[Join[Figure5DataTopRaw, extraPoints]];
Export["Figure5Top.csv", Figure5DataTop, "CSV"];
```

```
figure5DataBottomRaw =
  Table[{t, vmlif[warmBeamCool[2, t]], vml2if[warmBeamCool[2, t]]}, {t, 0, 4750}];
```

```
extraPoints = {#, vmlif[warmBeamCool[2, #]], vml2if[warmBeamCool[2, #]]} & /@
   {endwpoint, endwpoint + endb};
```

In[299]:=
```
figure5DataBottom = Sort[Join[figure5DataBottomRaw, extraPoints]];
Export["Figure5Bottom.csv", figure5DataBottom, "CSV"]
```

Out[300]= Figure5Bottom.csv

In[120]:=
```
Export["Figure_5.eps", pg]
```

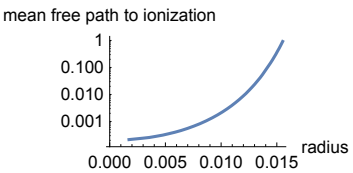
Out[120]= Figure_5.eps

Equation 2: Estimate the ionization radius for a Li atom.

Estimate ionization radius in the Magnum-PSI plasma beam by finding a radius such that (Equation 2 in the paper).

In[89]:=
```
data = Import[NotebookDirectory[] <> "Chandra_li_mfp_figure.csv", "CSV"];
interpolation = Interpolation[data];
maxInt = Max[interpolation[[3]]];
ListLogPlot[data, Joined → True, ImageSize → Small,
 AxesLabel → {"radius", "mean free path to ionization"}]
rLiIonization = 0.009 35;
(* found by trial and error. Verify below that the integral of mean-
 free-paths out to the largest given radius is (nearly) 1. *)
NIntegrate[1/interpolation[x], {x, rLiIonization, maxInt}]
```

Out[92]=



Out[94]= 1.008 8